ETSI TS 102 690 V1.2.1 (2013-06)

Machine-to-Machine communications (M2M);
Functional architecture

Reference
RTS/M2M-00002ed121

Keywords
architecture, M2M

ETSI

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - NAF 742 C
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° 7803/88

Important notice

Individual copies of the present document can be downloaded from:
http://www.etsi.org

The present document may be made available in more than one electronic version or in print. In any case of existing or perceived difference in contents between such versions, the reference version is the Portable Document Format (PDF). In case of dispute, the reference shall be the printing on ETSI printers of the PDF version kept on a specific network drive within ETSI Secretariat.

Users of the present document should be aware that the document may be subject to revision or change of status. Information on the current status of this and other ETSI documents is available at
http://portal.etsi.org/tb/status/status.asp

If you find errors in the present document, please send your comment to one of the following services:
http://portal.etsi.org/chaircor/ETSI_support.asp

Copyright Notification

Contents

Intellectual Property Rights

IPRs essential or potentially essential to the present document may have been declared to ETSI. The information pertaining to these essential IPRs, if any, is publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI Web server (http://ipr.etsi.org).

Pursuant to the ETSI IPR Policy, no investigation, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

Foreword

This Technical Specification (TS) has been produced by ETSI Technical Committee Machine-to-Machine communications (M2M).

1 Scope

The present document describes the overall end to end M2M functional architecture, including the identification of the functional entities and the related reference points.

The M2M functional architecture is designed to make use of an IP capable underlying network including the IP network service provided by 3GPP, TISPAN and 3GPP2 compliant systems. The use of other IP capable networks is not intentionally excluded. Non IP data services are used in the context of the present document for the purpose of out of band communication. Details of such usage are not depicted in the present document.

2 References

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the reference document (including any amendments) applies.

Referenced documents which are not found to be publicly available in the expected location might be found at http://docbox.etsi.org/Reference.

NOTE: While any hyperlinks included in this clause were valid at the time of publication ETSI cannot guarantee their long term validity.

2.1 Normative references

The following referenced documents are necessary for the application of the present document.

[1] ETSI TS 102 921: "Machine-to-Machine communications (M2M); mIa, dIa and mId interfaces".

[2] IETF RFC 6267: "MIKEY-IBAKE: Identity-Based Authenticated Key Exchange (IBAKE) Mode of Key Distribution in Multimedia Internet KEYing (MIKEY)".

[3] IETF RFC Draft: "An EAP Authentication Method Based on Identity-Based Authenticated Key Exchange".

NOTE: Available at http://tools.ietf.org/html/draft-cakulev-emu-eap-ibake.

[4] IETF RFC Draft: "IBAKE: Identity-Based Authenticated Key Exchange".

NOTE: Available at http://tools.ietf.org/html/draft-cakulev-ibake.

[5] ETSI TS 124 109: "Universal Mobile Telecommunications System (UMTS); LTE; Bootstrapping interface (Ub) and network application function interface (Ua); Protocol details (3GPP TS 24.109)".

[6] OMA-AD-DM-V1-3 (Version 1.3): "Device Management Architecture".

[7] OMA-TS-DM-Notification-V1-3 (Version 1.3): "Device Management Notification Initiated Session".

[8] OMA-TS-DM-Protocol-V1-3 (Version 1.3): "Device Management Protocol".

[9] OMA-TS-DM-Sessionless-V1-3 (Version 1.3): "Device Management Sessionless Message".

[10] Broadband Forum TR-069: "CPE WAN Management Protocol" Version 1.3, Issue: 1 Amendment 4. Issue Date: July 2011.

NOTE: Available at http://www.broadband-forum.org/technical/download/TR-069_Amendment-4.pdf

[11] OMA-TS-MLP-V3-3-20110719-A (Version 3.3): "Mobile Location Protocol 3.3".

[12] ETSI TS 123 271: "Digital cellular telecommunications system (Phase 2+); Universal Mobile Telecommunications System (UMTS); LTE; Functional stage 2 description of Location Services (LCS) (3GPP TS 23.271)".

[13] OMA-TS-DM-StdObj-V1-3-20101207-C (Draft Version 1.3): "OMA Device Management Standardized Objects".

[14] Broadband Forum TR-106: "Data Model Template for TR-069-Enabled Devices, Issue: 1 Amendment 6".

[15] OMA-TS-DM-TND-V1-3 (Version 1.3): "Device Management Tree and Description".

[16] OMA-TS-DCMO-V1-0 (Version 1.0): "Device Capability Management Object".

[17] Broadband Forum TR-157: "Component Objects for CWMP, Issue: 1 Amendment 3".

[18] OMA-TS-DiagMonFunctions-1-0 (Version 1.0): "DiagMon Functions Supplemental Specification".

[19] Broadband Forum TR-181: "Device Data Model for TR-069", Issue 2 Amendment 2.

[20] OMA-TS-LAWMO-V1-0 (Version 1.0): "Lock and Wipe Management Object".

[21] OMA-TS-DiagMonTrapMOFrame-V1-2 (Version 1.2): "Diagnostics and Monitoring Trap Framework Management Object".

[22] OMA-TS-DiagMonTrapEvents-V1-2 (Version 1.2): "Diagnostics and Monitoring Trap Events Specifications".

[23] OMA-TS-DiagMonMO-V1-0 (Version 1.0): "Diagnostics and Monitoring Management Object".

[24] OMA-TS-DM-FUMO-V1-0 (Version 1.0): "Firmware Update Management Object".

[25] OMA-TS-DM-SCOMO-V1-0 (Version 1.0): "Software Component Management Object".

[26] OMA-TS-GwMO-V1-0 (Version 1.0): "A Gateway Management Object technical Specification".

[27] ETSI TS 102 671 (V9.0.0): "Smart Cards; Machine to Machine UICC; Physical and logical characteristics (Release 9)".

[28] ETSI TS 102 310 (V9.0.0): "Smart Cards; Extensible Authentication Protocol support in the UICC (Release 8)".

[29] ETSI TS 133 220: "Digital cellular telecommunications system (Phase 2+); Universal Mobile Telecommunications System (UMTS); LTE; Generic Authentication Architecture (GAA); Generic Bootstrapping Architecture (GBA) (3GPP TS 33.220)".

[30] ETSI TS 187 003 (V2.3.2): "Telecommunications and Internet Converged Services and Protocols for Advanced Networking (TISPAN); NGN Security; Security Architecture".

[31] ETSI TS 102 484: "Smart Cards; Secure channel between a UICC and an end-point terminal (Release 9)".

[32] IETF RFC 2046: "Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types".

[33] IETF RFC 2560: "X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP".

[34] IETF RFC 2663: "IP Network Address Translator (NAT) Terminology and Considerations".

[35] IETF RFC 2716: "PPP EAP TLS Authentication Protocol".

[36] IETF RFC 4366: "Transport Layer Security (TLS) Extensions".

[37] IETF RFC 3748: "Extensible Authentication Protocol (EAP)".

[38] IETF RFC 3986: "Uniform Resource Identifier (URI): Generic Syntax".

[39] IETF RFC 4186: "Extensible Authentication Protocol Method for Global System for Mobile Communications (GSM) Subscriber Identity Modules (EAP-SIM)".

[40] IETF RFC 4187: "Extensible Authentication Protocol Method for 3rd Generation Authentication and Key Agreement (EAP-AKA)".

[41] IETF RFC 4346: "The Transport Layer Security (TLS) Protocol Version 1.1".

[42] IETF RFC 5216: "The EAP-TLS Authentication Protocol".

[43] IETF RFC 5246: "The Transport Layer Security (TLS) Protocol Version 1.2".

[44] IETF RFC 6066: "Transport Layer Security (TLS) Extensions: Extension Definitions".

[45] ETSI TS 131 102: "Universal Mobile Telecommunications System (UMTS); LTE; Characteristics of the Universal Subscriber Identity Module (USIM) application (3GPP TS 31.102)".

[46] IETF RFC 5448: "Improved Extensible Authentication Protocol Method for 3rd Generation Authentication and Key Agreement (EAP-AKA')".

[47] IETF RFC 5487: "Pre-Shared Key Cipher Suites for TLS with SHA-256/384 and AES Galois Counter Mode".

[48] IETF RFC 5191: "Protocol for Carrying Authentication for Network Access (PANA)".

[49] IETF RFC 5280: "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile".

[50] IETF RFC 4279: "Pre-Shared Key Ciphersuites for Transport Layer Security (TLS)".

[51] 3GPP2 S.S0109-0: "Generic Bootstrapping Architecture (GBA) Framework".

[52] ETSI TS 129 109: "Digital cellular telecommunications system (Phase 2+); Universal Mobile Telecommunications System (UMTS); LTE; Generic Authentication Architecture (GAA); Zh and Zn Interfaces based on the Diameter protocol; Stage 3 (3GPP TS 29.109)".

[53] IETF RFC 2045: "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies".

2.2 Informative references

The following referenced documents are not necessary for the application of the present document but they assist the user with regard to a particular subject area.

[i.1] Fielding, Roy Thomas (2000): "Architectural Styles and the Design of Network-based Software Architectures", Doctoral dissertation, University of California, Irvine.

NOTE: Available at Architectural Styles and the Design of Network-based Software Architectures.

[i.2] 3GPP TR 23.888: "System Improvements for Machine-Type Communications (MTC)".

[i.3] IEEE 802.15.1-2005: "IEEE Standard for Information technology - Telecommunications and information exchange between systems - Local and metropolitan area networks - Specific requirements. Part 15.1: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Wireless Personal Area Networks (WPANs)".

[i.4] IEEE 802.1X-2010: "IEEE Standard for Local and metropolitan area networks--Port-Based Network Access Control".

[i.5] IEEE 802.16e-2005: "IEEE Standard for Local and Metropolitan Area Networks - Part 16: Air Interface for Fixed and Mobile Broadband Wireless Access Systems - Amendment for Physical and Medium Access Control Layers for Combined Fixed and Mobile Operation in Licensed Bands".

[i.6] ETSI TR 102 725: "Machine to Machine Communications (M2M); Definitions".

NOTE: Explanations on the acronyms and abbreviations used in the present document can be found in TR 102 725.

3 Definitions, symbols and abbreviations

3.1 Definitions

For the purposes of the present document, the terms and definitions given in TR 102 725 [i.6] apply.

3.2 Symbols

For the purposes of the present document, the symbols given in TR 102 725 [i.6] apply.

3.3 Abbreviations

For the purposes of the present document, the abbreviations given in TR 102 725 [i.6] apply.

4 High level architecture

Figure 4.1 provides a High-level architecture for M2M.

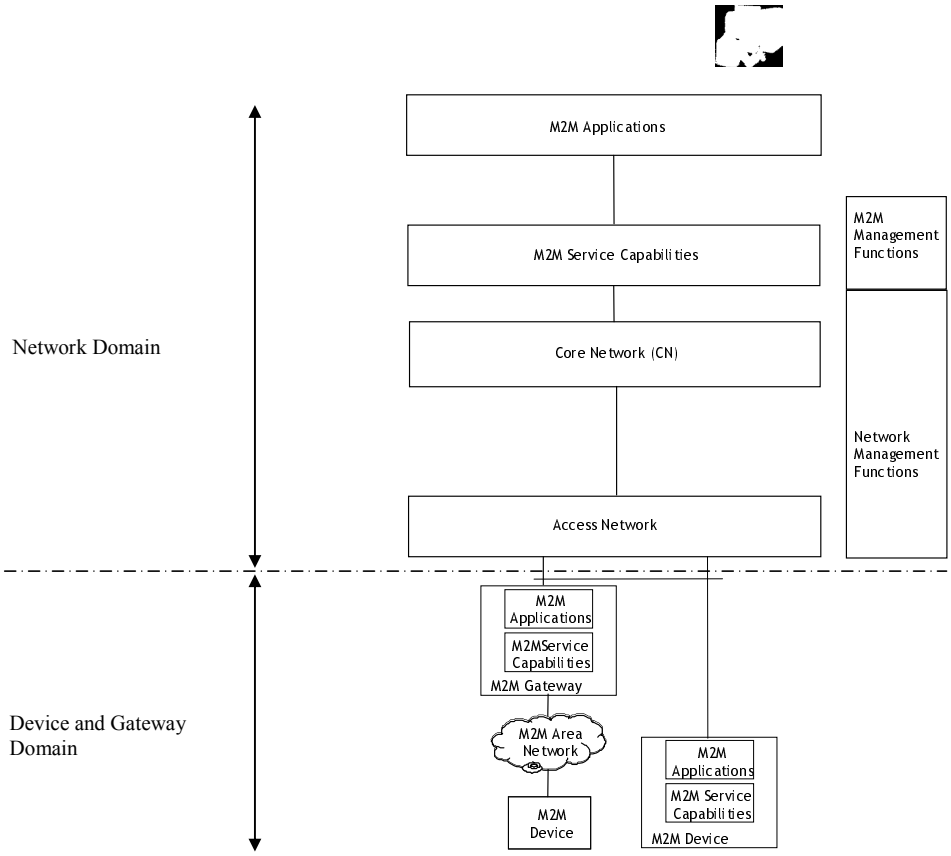


Figure 4.1: High level architecture for M2M

The High level architecture for M2M includes a Device and Gateway Domain and a Network domain.

- The Device and Gateway Domain is composed of the following elements:
  - **M2M Device:** A device that runs M2M Application(s) using M2M Service Capabilities. M2M Devices connect to Network Domain in the following manners:
    - **Case 1 "Direct Connectivity":** M2M Devices connect to the Network Domain via the Access network. The M2M Device performs the procedures such as registration, authentication, authorization, management and provisioning with the Network Domain. The M2M Device may provide service to other devices (e.g. legacy) connected to it that are hidden from the Network Domain.
    - **Case 2 "Gateway as a Network Proxy":** The M2M Device connects to the Network Domain via an M2M Gateway. M2M Devices connect to the M2M Gateway using the M2M Area Network. The M2M Gateway acts as a proxy for the Network Domain towards the M2M Devices that are connected to it. Examples of procedures that are proxied include: authentication, authorization, management and provisioning.

    M2M Devices may be connected to the Networks Domain via multiple M2M Gateways.
  - **M2M Area Network:** provides connectivity between M2M Devices and M2M Gateways.

    Examples of M2M Area Networks include: Personal Area Network technologies such as IEEE 802.15.1 [i.3], Zigbee®, Bluetooth®, IETF ROLL, ISA100.11a, etc. or local networks such as PLC, M-BUS, Wireless M-BUS and KNX.
  - **M2M Gateway:** A gateway that runs M2M Application(s) using M2M Service Capabilities. The Gateway acts as a proxy between M2M Devices and the Network Domain. The M2M Gateway may provide service to other devices (e.g. legacy) connected to it that are hidden from the Network Domain.

    As an example an M2M Gateway may run an application that collects and treats various information (e.g. from sensors and contextual parameters).

The Network Domain is composed of the following elements:

- **Access Network:** Network which allows the M2M Device and Gateway Domain to communicate with the Core Network.

  Access Networks include (but are not limited to): xDSL, HFC, satellite, GERAN, UTRAN, eUTRAN, W-LAN and WiMAX.
- **Core Network:** provides:
  - IP connectivity at a minimum and potentially other connectivity means.
  - Service and network control functions.
  - Interconnection (with other networks).
  - Roaming.
  - Different Core Networks offer different features sets.
  - Core Networks (CNs) include (but are not limited to) 3GPP CNs, ETSI TISPAN CN and 3GPP2 CN.
- **M2M Service Capabilities:**
  - Provide M2M functions that are to be shared by different Applications.
  - Expose functions through a set of open interfaces.
  - Use Core Network functionalities.
  - Simplify and optimize application development and deployment through hiding of network specificities.
- **M2M applications:** Applications that run the service logic and use M2M Service Capabilities accessible via an open interface.

- **Network Management Functions:** consists of all the functions required to manage the Access and Core networks: these include Provisioning, Supervision, Fault Management, etc.

- **M2M Management Functions:** consists of all the functions required to manage M2M Service Capabilities in the Network Domain. The management of the M2M Devices and Gateways uses a specific M2M Service Capability.

    - The set of M2M Management Functions include a function for M2M Service Bootstrap (explained in clause 8.3). This function is called **MSBF** (M2M Service Bootstrap Function) and is realized within an appropriate server. The role of MSBF is to facilitate the bootstrapping of permanent M2M service layer security credentials in the M2M Device (or M2M Gateway) and the M2M Service Capabilities in the Network Domain.

    - Permanent security credentials that are bootstrapped using MSBF (such as the M2M Root Key, explained in clause 8.2.2) are stored in a safe location, which is called M2M Authentication Server (**MAS**). Such a server can be a AAA server. MSBF can be included within MAS, or may communicate the bootstrapped security credentials to MAS, through an appropriate interface (e.g. Diameter for the case where MAS is a AAA server). The corresponding permanent security credentials established in the D/G M2M Node during the bootstrap are stored in a Secured Environment Domain of the D/G M2M Node.

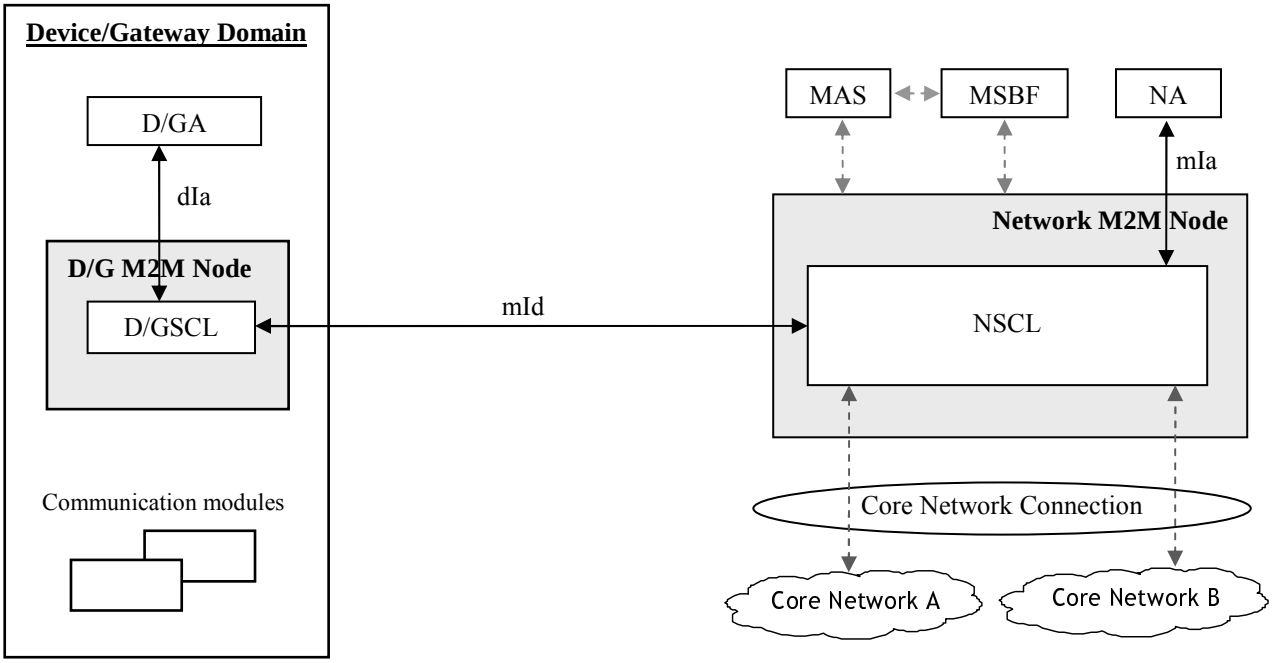
5 Functional architecture

This clause provides an overview of the M2M Service Capabilities and a description of the reference points.

5.1 Framework

5.1.1 Functions and reference points

This clause provides the framework of the functional architecture as well as reference points.



NOTE: Reference points shown with dashed lines denotes interfaces that are out of scope of the present document.

Figure 5.1: M2M Service Capabilities functional architecture framework

M2M Service Capabilities (SCs) Layer provides functions that are to be exposed on the reference points. M2M SCs can use Core Network functionalities through a set of exposed interfaces (e.g. existing interfaces specified by 3GPP, 3GPP2, ETSI TISPAN, etc.). Additionally, M2M SCs can interface to one or several Core Networks.

In the remaining of the present document the following terms will be used:

- NSCL: Network Service Capabilities Layer refers to M2M Service Capabilities in the Network Domain.
- GSCL: Gateway Service Capabilities Layer refers to M2M Service Capabilities in the M2M Gateway.
- DSCL: Device Service Capabilities Layer refers to M2M Service Capabilities in the M2M Device.
- SCL: Service Capabilities Layer, refers to any of the following: NSCL, GSCL, DSCL.
- D/GSCL: refers to any of the following: DSCL, GSCL.

The list of M2M Service Capabilities is provided below:

- Application Enablement (xAE);
- Generic Communication (xGC);
- Reachability, Addressing and Repository (xRAR);
- Communication Selection (xCS);
- Remote Entity Management (xREM);

- SECurity (xSEC);
- History and Data Retention (xHDR);
- Transaction Management (xTM);
- Compensation Broker (xCB);
- Telco Operator Exposure (xTOE);
- Interworking Proxy (xIP);

where x:

- N for Network.
- G for Gateway.
- D for Device.
- The present document supports the following M2M Service Capabilities: xAE, xGC, xSEC, xRAR, xREM, xTOE.

Not all M2M Service Capabilities are foreseen to be instantiated in the different parts of the system.

The M2M Service Capabilities above provide recommendations of logical grouping of functions, but does not mandate an implementation for M2M Service Capabilities Layer. The M2M Service Capabilities are therefore, not represented as separate entities in the message flows. However, the external reference points (mIa, mId, dIa) are mandated and are required for ETSI M2M compliance. In clauses 5.2 through 5.4 the description of the M2M Service Capabilities is informative. The Description of the reference points is normative.

M2M Node: is a logical representation of the M2M components in the M2M Device, M2M Gateway, or the M2M Core. An M2M Node shall include one SCL, and optionally an M2M Service Bootstrap function and an M2M Service Connection function. An M2M Node relies on a Secured Environment Domain, controlled by the M2M Service Provider associated with the SCL, to protect Sensitive Functions and Sensitive Data.

A Device/Gateway M2M Node shall be instantiated upon pre-provisioning or executing an M2M Service Bootstrap procedure on the M2M Device/Gateway with an M2M Service Provider. Each Device/Gagteway M2M Node may be instantiated with only one M2M Service Provider.

M2M Applications: are respectively Device Application (DA), Gateway Application (GA) and Network Application (NA). DA could reside in an M2M Device which implements M2M Service Capabilities (referred to in clause 6.1 as D device) or alternatively reside in an M2M Device which does not implement M2M Service Capabilities referred to in clause 6.1 as D' device).

mIa Reference Point:

allows a NA to access the M2M Service Capabilities in the Network Domain. The mIa Reference Point shall comply to the following specification [1].

dIa Reference Point:

allows a DA residing in an M2M Device to access the different M2M Service Capabilities in the same M2M Device or in an M2M Gateway;

- allows a GA residing in an M2M Gateway to access the different M2M Service Capabilities in the same M2M Gateway.

The dIa Reference Point shall comply to the following specification [1].

mId Reference Point:

allows an M2M Service Capabilities residing in an M2M Device or M2M Gateway to communicate with the M2M Service Capabilities in the Network Domain and vice versa. mId uses core network connectivity functions as an underlying layer. The mId Reference Point shall comply to the following specification [1].

5.1.2 High level flow of events

This clause provides a description of the high level flow of events when using SCLs to establish M2M Communications. This flow is depicted in Figure 5.2.

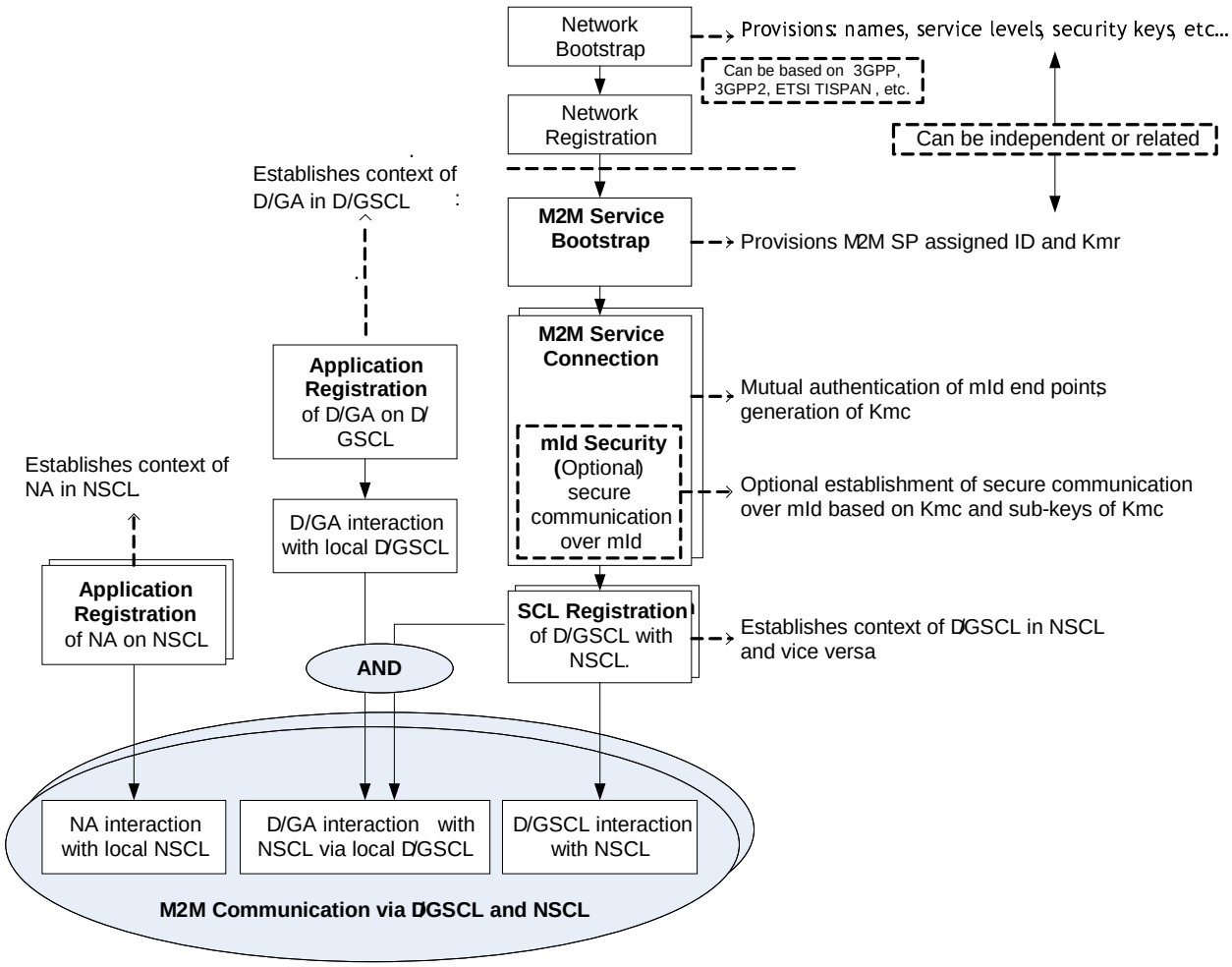


Figure 5.2: High level flow of events

M2M Service Bootstrap procedure between the D/G M2M Node (see clause 8.3) containing D/GSCL, MSBF and MAS (via Network M2M Node) is the mechanism by which Service Capability Layer credentials, such as permanent identifiers and the M2M Root Key are provisioned to D/G M2M Node (see clauses 7 and 8)) and M2M Authentication Server (MAS). These credentials are used for mutual authentication and secure communication between the D/GSCL on the D/G M2M Node and M2M Service Capability Layer in the network (NSCL), as well as authorization to access specific M2M services, and related accounting/billing functionality.

"M2M Service Connection procedure" between D/G M2M Node and Network M2M Node is the mechanism by which:

- mutual authentication of the mId end points and key agreement is performed. Security keys are generated as per the key hierarchy described in clause 8.2.2;
- a M2M Service Connection session is established between mId end points that is optionally encrypted based on the agreed key.

"SCL Registration" procedure of D/GSCL with NSCL is the mechanism by which:

- the D/GSCL on the D/G M2M Node registers with a M2M Service Capability Layer in the network (NSCL) in order to be able to use M2M services offered by the NSCL. A pre-requisite for D/GSCL Registration with M2M NSCL is a M2M Service Connection that has been established by performing the "M2M Service Connection procedure".

"Application Registration" procedure is the mechanism by which:

- the D/GA on a M2M Device or M2M Gateway register locally with the D/GSCL in order to use M2M services offered by the D/GSCL. The same holds for registration of NA on the NSCL. For the purpose of application-level authentication and encryption, application specific keys can be generated optionally according to the key hierarchy described in clause 8.2.2.

All the above steps are described in more details below:

The M2M Service Bootstrap and following M2M Service Connection and SCL registration procedures are preceded by Network Bootstrap and Network Registration.

Application Registration: involves local registration of an Application with the Local SCL. The purpose of Application registration is to allow the Application to use M2M services offered by the Local SCL. As a result of successful Application Registration, the SCL obtains context information on the registered application, see clause 9.3.2.8.2. If two applications register to a common Local SCL, then those applications can communicate via that Local SCL independently of other procedures. However, for an Application on one SCL to communicate with an Application registered to another SCL, other procedures are required to enable M2M communication between those SCLs.

Network Bootstrap: configures the M2M Device or the M2M Gateway with the initial configuration data that is necessary to connect and register to the Access Network (mobile or fixed). Examples of Bootstrap include:

- Bootstrap from UICC: if the M2M Device/Gateway is equipped with a UICC, then UICC is configured with all the necessary information for performing Access Network registration.

- Bootstrap OTA (Over the Air): the access credentials including the key material needed for registration operations is provisioned via an over-the-air mechanism.

Network Bootstrapping is outside the scope of the present document.

Network Registration: involves registration of the M2M Device/Gateway with the Access Network, based on the corresponding access network standards. As an example, in 3GPP networks during network registration the M2M Device/Gateway is mutually authenticated with the Access Network, and the two ends agree on a set of security keys for that access network session. In addition, registration involves IP address assignment, authorization approval for using specific Access Network services, and initiation of potential Access Network accounting operations. Network Registration is outside the scope of the present document.

M2M Service Bootstrap procedure: involves the provisioning of permanent M2M service credentials (e.g. identities, M2M Root Key), which will be used for connecting and registering D/GSCL with the NSCL. M2M Service Bootstrap procedure can optionally provision a list of one or more NSCL Identifiers that the D/GSCL uses as the next point of contact. If the M2M service credentials have been pre-provisioned (e.g. in UICC), no M2M Service Bootstrap procedure is needed. Otherwise, depending on whether a business relationship exists between the Access Network provider and the M2M Service Provider, as per clause 8.3, and based on the associated business relationship policies, M2M Service Bootstrap procedure is performed in one of the following ways:

- Bootstrap assisted by the Access Network layer: in this case, the Access Network provider and M2M service provider share a business relationship. In this scenario, M2M service layer credentials can be bootstrapped from the access network layer (e.g. from UICC, if available). Clause 8.3.2 provides the details of such a procedure.

- Bootstrap without assistance from the access network: Details of this bootstrap mechanism are provided in clause 8.3.3. Potential reasons for this case could be the following:

    - There are no business relationships between the access Network provider and the M2M service provider.

    - The access network only facilitates insecure transport of M2M traffic, i.e. it does not provide authentication, key agreement, encryption or integrity protection of M2M information. Example of such a case is an access network that is simply used for M2M connectivity, without providing any secure transport connections.

In the optional case where a M2M Device/Gateway supports device Integrity Validation, device Integrity Validation procedures are performed prior to executing the M2M Service Bootstrap procedures.

M2M Service Connection procedure: Operations included in this procedure are:

- Mutual authentication of mId end points.
- Optional Agreement on M2M Connection Key (Kmc), see clause 8.2.2.
- optionally establishment of a secure session via mId using encrypted communication.

Details of M2M Service Connection procedures between D/G M2M Node and Network M2M Node are defined in clause 8.4.

SCL Registration of D/GSCL with NSCL: involves registration of the D/GSCL on a D/G M2M Node with the M2M Service Capability Layer on the network (NSCL). Details on this procedure are provided in clause 9.3.2.6.2.
A pre-requisite for performing SCL Registration of D/GSCL with NSCL is successful M2M Service Connection between D/G M2M Node and Network Node. D/GSCL registration update procedures (see clause 9.3.2.6.4) take place either periodically, or on demand by the D/GSCL or the NSCL. The frequency of D/GSCL registration updates is decided by the M2M Service Provider. The time interval between two consecutive D/GSCL registration update procedures is larger than the interval between two reachability verification procedures. As a result of successful D/GSCL registration, the D/GSCL and NSCL exchange context information, see clause 9.3.2.6.2, as well as initiation of accounting operations is performed.

Once M2M Service Bootstrap procedure, M2M Service Connection procedure, D/GSCL as well as D/GA and/or NA Registration procedures are performed, the following procedures can be executed:

- RESTful procedures for Access rights management, Container management, Group management, Resource discovery, Collection management, Subscription management, Announce/De-announce etc (see clause 9.3.2);
- Remote Entity Management (xREM) procedures (see description of NREM, GREM, DREM M2M Service Capabilities in clause 5, description of <*mgmtObj*> resource in clause 9.2.3.27 and annex B).

The actual communication between any combination of NA and D/GA is established by use of these sets of procedures via interaction with NSCL and D/GSCL.

5.2 M2M Service Capabilities in the Network Domain

5.2.1 Network Application Enablement (NAE) capability

The NAE Capability in the NSCL is the single contact point to NAs. The NAE Capability provides the following functionalities:

- Exposes functionalities implemented in NSCL via a single reference point: mIa.
- Allows NA to register to the NSCL.
- Performs routing between NAs and capabilities in the NSCL. Routing is defined as the mechanism by which a specific request is sent to a particular capability or an instance of that capability when e.g. load balancing is implemented.
- Allows routing towards different capabilities.
- Generates charging records pertaining to the use of capabilities.

5.2.2 Network Generic Communication (NGC) capability

The NGC capability in the NSCL is the single point of contact for communication with GSCL and DSCL. It provides the following functionalities:

- Provides transport session establishment and teardown along with security key negotiation. If such a transport session establishment is performed in a secure way, then the security key negotiation uses keys as provided by NSEC.

- Provides encryption/integrity protection on data exchanged with the M2M Devices/Gateways, see clause 8.5. Key material for encryption and integrity protection is derived upon secure session establishment.
- Ensures secure delivery of application data from/to M2M Gateways or M2M Devices and the NSCL when security is required.
- Reports transmission errors.
- Is optionally able to inspect traffic generated by a particular M2M Device or M2M Gateway and verify it is matching a given traffic pattern (e.g. number of connections per day, traffic volume per day, more than 20 % of the monthly average traffic is generated in one day, etc). Other policies pertaining to traffic inspection are possible.

5.2.3 Network Reachability, Addressing and Repository (NRAR) Capability

The NRAR Capability in the NSCL provides the following functionalities:

- Provides a mapping between the name of a M2M Device or an M2M Gateway or a group of M2M Devices/M2M Gateways and a set of information:
  - Set of routable network addresses of the M2M Device or M2M Gateway (e.g. in case of GPRS network, a M2M Device or M2M Gateway can have 2 addresses: an IP address and a MSISDN).
  - Reachability status of an M2M Device or M2M Gateway.
  - Scheduling information pertaining to reachability of the M2M Device or M2M Gateway, if available.
- Manages subscriptions and notifications pertaining to events.
- Allows to create, delete and list a group of M2M Devices or M2M Gateways.
- Store NA, DSCL, GSCL related registration information.
- Store application (NA, DA, GA) and SCL data and make it available, on request or based on subscriptions, subject to access rights and permissions.
- Informs NSCL about reachability and scheduling information pertaining to reachability of the M2M Gateway.

5.2.4 Network Communication Selection (NCS) Capability

The NCS Capability in the NSCL provides the following functionalities:

- Provides network selection, based on policies, when the M2M Device or M2M Gateway can be reached through several networks or several bearers.
- Provides alternative Network or Communication Service selection after a communication failure using a first selected Network or Communication Service.

5.2.5 Network Remote Entity Management (NREM) Capability

The NREM Capability in the NSCL provides the following functionalities:

- Provides Configuration Management (CM) functions. CM is the means to provision a set of Management Objects in an M2M Device, an M2M Gateway, a set of M2M Devices or a set of M2M Gateways. Examples of CM include: the Activation of Fault Management (FM) and Performance Management (PM) collection.
- Collects and stores Performance Management (PM) data, e.g. radio interference management data. Provides upon request or other policies the related PM data to NAs and/or M2M Management Functions.
- Collects and stores Fault Management (FM) data. Provides upon request or other policies the related FM data to NAs and/or M2M Management Functions.

- Triggers connection establishment.
- Performs software and firmware upgrade of M2M Device or M2M Gateways:
  - The NREM supports independent management of M2M software and M2M firmware for different part of a M2M Device or M2M Gateway by different management authorities. The management authorities supported in a M2M Device or M2M Gateway are:
    - One M2M Service Provider - for management of SCL related M2M software and M2M firmware.
    - One or more Application Providers (including the manufacturer) - for management of M2M software and M2M firmware that is controlling application specific hardware modules in the M2M Device/M2M Gateway.
  - If M2M Device or M2M Gateway integrity validation is supported, the NREM supports NSEC-triggered post-validation actions including remediation, roll-back or update of software and firmware of the M2M Device/M2M Gateways.
  - Information including integrity-protected trusted reference values, policy and configuration related information (i.e. identification of executables for integrity validation) can optionally be included in the post-validation entity-management procedures.
  - The NREM can optionally support Authentication and Authorization of a management authority as well as integrity checking of M2M software/M2M firmware from these management authorities.
  - The NREM is responsible for avoiding conflicts among multiple authorities.
- Supports several management protocols on the mId reference point (such as OMA-DM [8] and BBF TR-069 [10]) for managing different M2M Devices and M2M Gateways.
- Supports a common mechanism to indicate the management protocol to be used by NREM.
- Supports the following management functions at different layers of remote entities:
  - M2M application lifecycle management: installing, removing and upgrading applications in an M2M Device/M2M Gateway.
  - M2M service management: configuration management for the M2M Service Capabilities in the M2M Device/M2M Gateway.
  - M2M Area Network management: configuration management for the M2M Area Networks.
  - M2M device management: configuration management of the M2M Device/M2M Gateway.

5.2.6 Network Security Capability (NSEC)

The NSEC Capability in the NSCL provides the following functionalities:

- Supports M2M Service Bootstrap.
- Support key hierarchy realization for authentication and authorization.
- Performs mutual authentication and key agreement.
- Can verify the integrity validation status reported by the M2M Device/M2M Gateway and trigger appropriate post validation actions.
  - Further details on NSEC functionalities are provided in clause 8.2.3.1.

5.2.7 Network History and Data Retention (NHDR) capability

The NHDR Capability in the NSCL is an optional capability, i.e. deployed when needed/required by NSCL operator policies. The NHDR Capability provides the following functionalities:

- Archives relevant information pertaining to messages exchanged over the reference point and also internally to the NSCL based on policies.
- Interacts with the other capabilities residing in the NSCL to:
  - determine if and which information requires to be retained;
  - obtain the information to be stored from capability involved.

5.2.8 Network Transaction Management (NTM) capability

The NTM capability in the NSCL is an optional capability, i.e. deployed when needed/required by operator policies.

For the purpose of the present document, a transaction is an operation that involves several operations (from different entities in the M2M System) and requires atomicity in its execution. In case one single operation is not completed, the overall transaction is considered as unsuccessful and a rollback operation is triggered for operations that were reported to be successful.

The NTM Capabilities provides the following functionalities:

- propagates the individual operation requests;
- aggregates the results of the individual operations and commits the transaction when all individual operations have completed successfully;
- trigger a roll-back if any individual operation fails.

5.2.9 Network Interworking Proxy (NIP) capability

The NIP Capability in the NSCL is an optional capability, i.e. deployed when needed/required by operator policies.

The NIP Capability provides the following functionalities:

- Provides interworking between non ETSI compliant devices or gateways and the NSCL.

NOTE 1: Depending on the nature of existing non ETSI compliant M2M deployments, there might be other ways to provide interworking.

NOTE 2: It is recognized that full interworking is not possible in some cases, depending on the characteristics of the M2M Device/Gateway.

5.2.10 Network Compensation Brokerage (NCB) capability

The NCB Capability in the NSCL is an optional Capability, i.e. deployed when needed/required by operator policies.

Compensation involves three roles: a Customer (any application buying a service from a service provider), a Service Provider (any application providing the service) and a Broker implemented by the NCB Capability in the NSCL. Customers buy or acquire services from a Service Provider by applying electronic compensation. A Broker represents a trusted third party who will bill Customers according to their expenditures committed towards Service Providers, and accordingly redeems the Service Providers for such amounts.

The NCB Capability provides the following functionalities:

- Submits compensation tokens (i.e. electronic money) to requesting Customers.
- Verifies the validity of compensation tokens.
- Bills the Customer of compensation tokens for the amount spent.

- Redeems Service Providers for tokens acquired as compensation for services provided to customers.

5.2.11 Network Telco Operator Exposure (NTOE) Capability

The NTOE Capability in the NSCL is an optional Capability, i.e. deployed when needed/required by operator policies.

NTOE provides the following functionality:

- Interworking and using of Core Network services exposed by the Network Operator.

5.3 Service Capabilities in the M2M Gateway

5.3.1 Gateway Application Enablement (GAE) capability

The GAE Capability in the GSCL is the single contact point to GA and DA. The GAE Capability provides the following functionalities:

- Exposes functionalities implemented in GSCL via the single reference point: dIa.
- Allows GA and DA to register to the GSCL.
- Provides intra M2M Area Network transmission of a message sent by any application (GA or DA) to any other application (GA or DA) connected to the same M2M Gateway.
- Can provide authentication and authorization of GA and DA before allowing them to access a specific set of capabilities.

NOTE: Details of the methods used for authentication of GA and DA are out of scope of the present document.

- Checks if a specific request on dIa is valid before routing it to other M2M Service Capabilities. If a request is not valid an error is reported to the M2M Application (GA or DA).
- Can generate charging records pertaining to the use of capabilities.

The Secured Environment Domains for use by the GAE are controlled by the respective application providers to perform Sensitive Functions [i.6] pertaining to applications.

5.3.2 Gateway Generic Communication (GGC) capability

The GGC Capability is the single point of contact for communication with the NSCL. The GGC Capability provides the following functionalities:

- Provides transport session establishment and teardown along with security key negotiation. If such a transport session establishment is performed in a secure way, then the security key negotiation uses keys as provided by GSEC.
- Provides encryption/integrity protection on data exchanged with the NSCL, see clause 8.5. Key material for encryption and integrity protection is derived upon secure session establishment.
- Ensures secure delivery of application data from/to M2M Gateways and the NSCL when security is required.
- Reports transmission errors.

The GGC capability in the GSCL provides the following functionalities (through use of other capabilities):

- Relays messages received from the NGC Service Capability in the NSCL towards GSCL.
- Relays messages received from GSCL to the NGC capability in the NSCL.
- Delivers messages in accordance with the Service Class.
- Handles name resolution for requests originated within an M2M Area Network.

- Reports errors triggered by e.g.:
  - The identifier of the recipient is not existing.
  - The requested Service Class is not supported by the recipient.
  - Etc.
- Provides secure transport functions such as encryption and integrity protection. NGC and GGC are the two end-points of such security associations and sessions.
- Sensitive Functions [i.6] performed by the GGC run inside the Secured Environment Domain controlled by the owner of the involved key.

5.3.3 Gateway Reachability, Addressing and Repository (GRAR) capability

The GRAR Capability in the GSCL provides the following functionalities:

- Provides a mapping between the name of a M2M Device or a group of M2M Devices and a set of information:
  - Set of routable network addresses of the M2M Device.
  - Reachability status of an M2M Device.
  - Scheduling information pertaining to reachability of the M2M Device, if available.
- Manages subscriptions and notifications pertaining to events.
- Allows to create, delete and list a group of M2M Devices.
- Store GA and DA related registration information and NSCL information.
- Store application (NA, DA, GA) and SCL (GSCL, NSCL) data and make it available, on request or based on subscriptions, subject to access rights and permissions.

5.3.4 Gateway Communication Selection (GCS) capability

The GCS Capability in the GSCL provides the following functionalities:

- Provides network selection, based on policies, when the NSCL can be reached through several networks or several bearers.
- Provides alternative Network or Communication Service selection after a communication failure using a first selected Network or Communication Service.

5.3.5 Gateway Remote Entity Management (GREM) capability

The GREM Capability in the GSCL provides the following functionalities:

- Acts as a remote management client to perform the DREM functionalities as described in clause 5.4.5 for the M2M Gateway itself.
- Acts as a remote management proxy for M2M Devices connected to the managed M2M Area Network:
  - Accepts and processes management requests, targeted at one or multiple M2M Devices, from the NREM.
  - Accepts and processes management requests from one or multiple M2M Devices and/or further contacts the NREM on behalf of the M2M Device(s) (e.g. in the case of fault detection and reporting).
  - Requests the NREM to start performing M2M Device management tasks (e.g. firmware/software update, fault and performance diagnostics) with one or multiple M2M Devices.

- Initiates as per policies installed in the M2M Gateway interactions for device management tasks (e.g. firmware/software update, fault and performance diagnostics), with one or multiple M2M Devices.
- Informs the NREM of the results of the device management interactions.
- Receives and stores firmware/software and management data (including bootstrap related) files pertaining to the M2M Devices within the M2M area network.
- Supports process optimization to reduce the signalling and data traffic to/from the NREM in the case of managing multiple M2M Devices with the same operations (e.g. bulk firmware/software update, fault and performance diagnostics).
- Supports a common mechanism to indicate the management protocol used by M2M Device(s).
- Performs protocol translation if the management protocol in the M2M area network is different from the management protocol used over mId reference point.
- Schedules remote management tasks for sleeping M2M Devices.
- Supports the following management functions at different layers of remote entities:
  - M2M application lifecycle management: installing, removing and upgrading applications in an M2M Device.
  - M2M service management: configuration management for the M2M Service Capabilities in the M2M Device.
  - M2M Area Network management: configuration management for the M2M Area Networks.
  - M2M device management: configuration management of the M2M Device.

5.3.6 Gateway SECurity (GSEC) capability

The GSEC Capability in the GSCL provides the following functionalities:

- Supports M2M Service Bootstrap.
- Support key hierarchy realization for authentication and authorization.
- Initiates mutual authentication and key agreement.
- If supported by the M2M Gateway, can report the integrity validation status to the NSEC and react to post validation actions triggered by NSEC.
- The GSEC is responsible for the storage and handling of M2M Connection Keys (see clause 8.2.2). Sensitive Functions [i.6] performed by the GSEC shall run inside a Secured Environment Domain controlled by the M2M Service Provider owning the M2M Node.
  - Further details on GSEC functionalities are provided in clause 8.2.3.2.

5.3.7 Gateway History and Data Retention (GHDR) capability

The GHDR Capability in the GSCL is an optional capability, i.e. deployed when needed/required by policies. The GHDR Capability provides the following functionalities:

- Archives relevant information pertaining to messages exchanged over the reference point and also internally to the GSCL based on policies.
- Interacts with the other capabilities residing in the GSCL to:
  - determine if and which information requires to be retained;
  - obtain the information to be stored from capability involved.
- Can report to NHDR history data stored information in the GHDR.

5.3.8 Gateway Transaction Management (GTM) capability

The GTM capability in the GSCL is an optional capability, i.e. deployed when needed/required by policies.

The GTM Capabilities provides the following functionalities:

- propagates the individual operation requests;
- aggregates the results of the individual operations and commits the transaction when all individual operations have completed successfully;
- trigger a roll-back if any individual operation fails.

5.3.9 Gateway Interworking Proxy (GIP) capability

The GIP Capability in the M2M Gateway is an optional capability, i.e. deployed when needed/required by policies.

The GIP Capability provides the following functionalities:

- Provides interworking between non ETSI compliant devices and the GSCL.
- Provides interworking with one or more M2M Area Networks and the GSCL.
- GIP communicates via reference point dIa with GSCL.

NOTE 1: Depending on the nature of existing non ETSI compliant M2M deployments, there might be other ways to provide interworking.

NOTE 2: It is recognized that full interworking is not possible in some cases, depending on the characteristics of the M2M Device/Gateway.

NOTE 3: GIP is considered an Application in the M2M Gateway.

5.3.10 Gateway Compensation Brokerage (GCB) capability

The GCB Capability in the GSCL is an optional Capability, i.e. deployed when needed/required by operator policies.

The GCB Capability provides the following functionalities:

- Submits compensation tokens (i.e. electronic money) to requesting Customers.
- Verifies the validity of compensation tokens.
- Bills the Customer of compensation tokens for the amount spent.
- Redeems Service Providers for tokens acquired as compensation for services provided to customers.

5.4 Service Capabilities in the M2M Device

5.4.1 Device Application Enablement (DAE) capability

The DAE Capability in the DSCL is the single contact point to DA. The DAE Capability provides the following functionalities:

- Exposes functionalities implemented in DSCL via the single reference point: dIa.
- Allows DA to register to the DSCL.
- The DAE can use Secured Environment Domains controlled by the respective application providers to perform Sensitive Functions [i.6] pertaining to applications.

5.4.2 Device Generic Communication (DGC) capability

The DGC Capability is the single point of contact for communication with the NSCL. The DGC Capability provides the following functionalities:

- Provides transport session establishment and teardown along with security key negotiation. If such a transport session establishment is performed in a secure way, then the security key negotiation uses keys as provided by DSEC.
- Provides encryption/integrity protection on data exchanged with the NSCL, see clause 8.5. Key material for encryption and integrity protection is derived upon secure session establishment.
- Ensures secure delivery of application data from/to M2M Devices and the NSCL when security is required.
- Reports transmission errors.
- Sensitive Functions [i.6] performed by the DGC run inside the Secured Environment Domain controlled by the owner of the involved key.

5.4.3 Device Reachability, Addressing and Repository (DRAR) capability

The DRAR Capability in the DSCL provides the following functionalities:

- Manages subscriptions and notifications pertaining to events.
- Allows to create, delete and list a group of M2M Devices.
- Store DA related registration information and NSCL information.
- Store application (NA, DA, GA) and SCL (DSCL, NSCL) data and make it available, on request or based on subscriptions, subject to access rights and permissions.
- Informs NSCL about reachability and scheduling information pertaining to reachability of the M2M Device.

5.4.4 Device Communication Selection (DCS) capability

The DCS Capability in the DSCL provides the following functionalities:

- Provides network selection, based on policies, when the NSCL can be reached through several networks or several bearers.
- Provides alternative Network or Communication Service selection after a communication failure using a first selected Network or Communication Service.

5.4.5 Device Remote Entity Management (DREM) capability

The DREM Capability in the DSCL provides the following functionalities:

- Performs CM, PM, FM and triggers connection establishment of the M2M Device itself.
- Performs software and firmware upgrade of M2M Device:
  - Supports independent management of M2M software and M2M firmware for different part of a M2M Device by different management authorities. The management authorities supported in a M2M Device are:
    - One M2M service provider - for management of SCL related M2M software and M2M firmware.
    - One or more application providers (including the manufacturer) - for management of M2M software and M2M firmware that is controlling application specific hardware modules in the M2M Device.

  - Can support post-validation actions including remediation, roll-back or update of software and firmware of the M2M Device in case of failures (including when triggered by the optional device Integrity Validation).
  - Information including integrity-protected trusted reference values, policy and configuration related information (i.e. identification of executables for integrity validation) can also be included in the post-validation entity-management procedures.
  - Can support Authentication and Authorization of a management authority as well as Integrity checking of M2M software/M2M firmware from these management authorities.
- Supports a common mechanism to indicate the management protocol to be used by the M2M Device.
- Supports the following management functions at different layers of remote entities:
  - M2M application lifecycle management: installing, removing and upgrading applications in an M2M Device.
  - M2M service management: configuration management for the M2M Service Capabilities in the M2M Device.
  - M2M device management: configuration management of the M2M Device.

5.4.6 Device SECurity (DSEC) capability

The DSEC Capability in the DSCL provides the following functionalities:

- Supports M2M Service Bootstrap.
- Support key hierarchy realization for authentication and authorization.
- Initiates mutual authentication and key agreement.
- If supported by the M2M Device, can report the integrity validation status to the NSEC and react to post validation actions triggered by NSEC.

The DSEC is responsible for the storage and handling of M2M Connection Keys (see clause 8.2.2). Sensitive Functions [i.6] performed by the DSEC shall run inside a Secured Environment Domain controlled by the M2M Service Provider owning the M2M Node.

Further details on DSEC functionalities are provided in clause 8.2.3.3.

5.4.7 Device History and Data Retention (DHDR) capability

The DHDR Capability in the DSCL is an optional capability, i.e. deployed when needed/required by policies. The DHDR Capability provides the following functionalities:

- Archives relevant information pertaining to messages exchanged over the reference point and also internally to the DSCL based on policies.
- Interacts with the other capabilities residing in the DSCL to:
  - determine if and which information requires to be retained;
  - obtain the information to be stored from capability involved.
- Can report to NHDR history data stored information in the DHDR.

5.4.8 Device Transaction Management (DTM) capability

The DTM capability in the DSCL is an optional capability, i.e. deployed when needed/required by policies.

The DTM Capabilities provides the following functionalities:

- propagates the individual operation requests;
- aggregates the results of the individual operations and commits the transaction when all individual operations have completed successfully;
- trigger a roll-back if any individual operation fails.

5.4.9 Device Interworking Proxy (DIP) capability

The DIP Capability in the M2M Device is an optional capability, i.e. deployed when needed/required by policies.

The DIP Capability provides the following functionalities:

- Provides interworking between non ETSI compliant devices and the DSCL.
- Provides interworking with one or more M2M Area Networks and the DSCL.
- DIP communicates via reference point dIa with DSCL

NOTE 1: Depending on the nature of existing non ETSI compliant M2M deployments, there might be other ways to provide interworking.

NOTE 2: It is recognized that full interworking is not possible in some cases, depending on the characteristics of the M2M Device/Gateway.

NOTE 3: DIP is considered an Application in the M2M Device.

5.4.10 Device Compensation Brokerage (DCB) capability

The DCB Capability in the DSCL is an optional Capability, i.e. deployed when needed/required by operator policies.

The DCB Capability provides the following functionalities:

- Submits compensation tokens (i.e. electronic money) to requesting Customers.
- Verifies the validity of compensation tokens.
- Bills the Customer of compensation tokens for the amount spent.
- Redeems Service Providers for tokens acquired as compensation for services provided to customers.

6 Reference points

6.1 Overview

This clause describes the mapping of reference points as described in clause 5.1 to the different deployment scenarios that are supported by the present document.

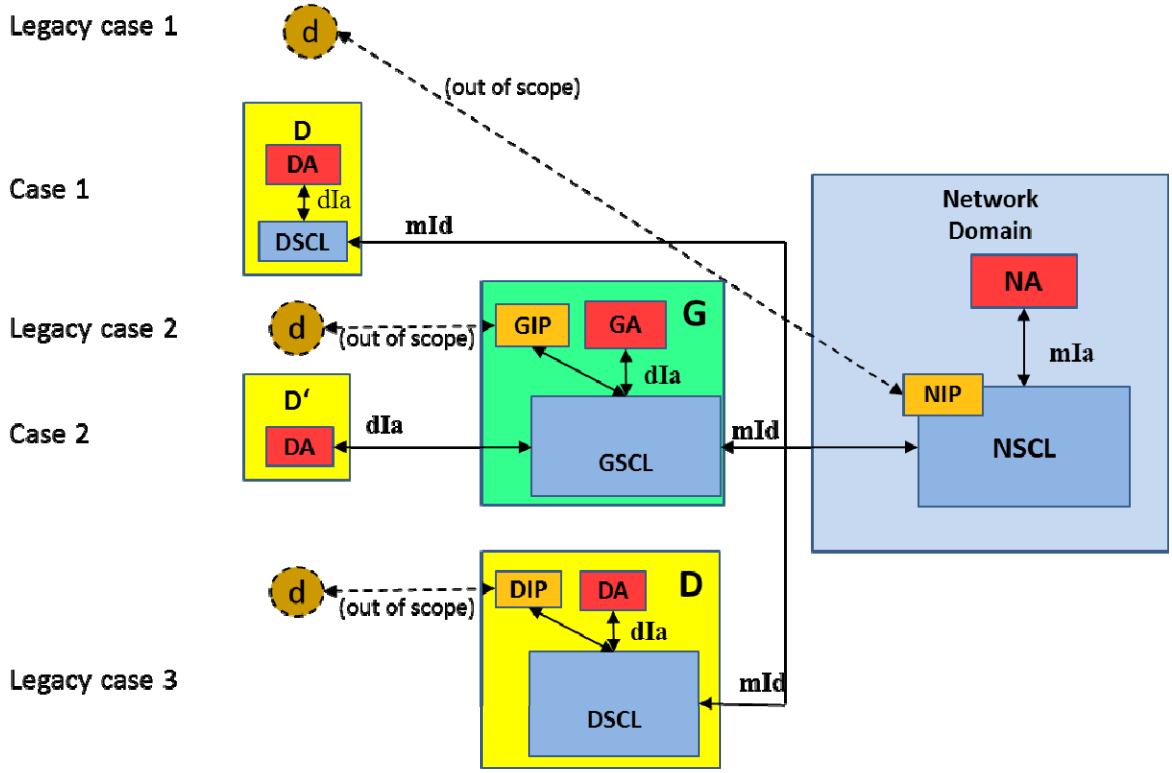


Figure 6.1: Mapping of reference points to different deployment scenarios

Gateway (G): shall provide M2M Service Capabilities (GSCL) that communicates to the NSCL using the mId reference point and to DA or GA using the dIa reference point.

Device (D): shall provide M2M Service Capability (DSCL) that communicates to an NSCL using the mId reference point and to DA using the dIa reference point.

Device' (D'): shall host DA that communicates to a GSCL using the dIa reference point. D' does not implement ETSI M2M Service Capabilities.

Additionally there is a non-ETSI M2M compliant device ('d') that connects to SCL using the xIP Capability (NIP, GIP, DIP). d devices do not use ETSI M2M defined reference points, however:

- GIP communicates via reference point dIa with GSCL.
- DIP communicates via reference point dIa with DSCL.

6.2 mIa

The mIa reference point offers generic and extendable mechanism for Network Applications interactions with the NSCL.

The mIa reference point, between NA and NSCL, shall support the procedures described in clause 9.2.3 and comply to [1]. These functions include:

- Registration of NA to the NSCL.
- Request to Read/Write, subject to proper authorization, information in the NSCL, GSCL, or DSCL.
- Request device management actions (e.g. software upgrade, configuration management).
- Subscription and notification to specific events.
- Request the creation, deletion and listing of group(s).

6.3 dIa

The dIa reference point offers generic and extendable mechanism for Device Application (DA)/Gateway Application (GA) interactions with the DSCL/GSCL.

The dIa reference point, between D/GA and DSCL/GSCL, shall support the procedures described in clause 9.2.3 and comply to [1]. These functions include:

- Registration of D/GA to GSCL.
- Registration of DA to DSCL.
- Request to Read/Write, subject to proper authorization, information in the NSCL, GSCL, or DSCL.
- Subscription and notification to specific events.
- Request the creation, deletion and listing of group(s).

6.4 mId

The mId reference point offers generic and extendable mechanism for SCL interactions.

The mId reference point, between SCLs, shall support the procedures described in clause 9.2.3 and comply to [1]. These functions include:

- Registration of a DSCL/GSCL to NSCL.
- Request to Read/Write, subject to proper authorization, information in the NSCL, GSCL, or DSCL.
- Request device management actions (e.g. software upgrade, configuration management).
- Subscription and notification to specific events.
- Request the creation, deletion and listing of group(s).
- Provides security related features as defined in clause 8.

7 M2M Identification and addressing

7.1 Introduction

This clause provides an overview of the identifiers that are used in the present document. It also specifies how identifiers are resolved into network addresses which are used for communication between entities in the M2M System.

The following provides the list of identifiers that are used in the present document:

- Application identifier, App-ID
- SCL identifier, SCL-ID
- M2M node identifier, M2M-Node-ID
- M2M Service Connection identifier, M2M-Connection-ID
- M2M Service Provider identifier, M2M-SP-ID
- MSBF identifier, MSBF-ID

7.2 M2M Identification

7.2.1 M2M Identifiers

7.2.1.1 Application Identifier

An Application Identifier, App-ID, uniquely identifies an M2M Application (NA, GA, DA) that is registered with a SCL. An App-ID shall identify an application for the purpose of interacting with the application. This identifier shall be globally unique. It is the responsibility of the M2M Service Provider to ensure that the App-ID is globally unique.

If multiple instances of the same M2M Application connects to the same SCL, then the App-ID of each instance shall be globally unique.

7.2.1.2 M2M Node Identifier

An M2M Node shall be identified by a globally unique identifier, the M2M-Node-ID. When instantiated in the M2M Device/Gateway, the M2M-Node-ID characteristics are summarized in Table 7.1.

7.2.1.3 SCL Identifier

A SCL shall be identified by a globally unique identifier, the SCL-ID. When instantiated in the M2M Device/Gateway, the SCL-ID characteristics are summarized in Table 7.1.

The M2M System shall allow the M2M Service Provider to set the SCL-ID and the M2M-Node-ID to the same value. This possibility could be limited by privacy issues.

7.2.1.4 M2M Service Connection Identifier

An M2M Service Connection is the connection between the M2M Device/Gateway and the NSCL and it is used for facilitating the SCLs and applications on both ends to communicate with each other. This connection relies on lower-layer connectivity at the physical, link, and network-layers. An M2M Service Connection shall be instantiated upon D/G SCL getting authenticated/authorized by a NSCL for connectivity. An M2M Service Connection can be torn down, re-established, and refreshed. An M2M Service Connection may also provide its own security (e.g. via TLS or IPsec).

An M2M Service Connection shall be identified by an M2M-Connection-ID whose characteristics a are summarized in Table 7.1.

7.2.1.5 M2M Service Provider Identifier

An M2M Service Provider shall be uniquely identified by the M2M Service Provider Identifier, M2M-SP-ID. This is a static value assigned to the Service Provider. M2M-SP_ID is used in the bootstrap procedure for key generation. More details can be found in [1].

7.2.1.6 MSBF Identifier

An MSBF shall uniquely identified by the MSBF Identifier, MSBF-ID. This is a static value assigned to the MSBF by the M2M Service Provider. MSBF-ID is used in the bootstrap procedure for key generation. More details can be found in [1].

7.2.2 M2M Identifiers lifecycle and characteristics

Table 7.1 provides for each M2M Identifier a format as well as other information pertaining to is assignment and lifecycle in general. This table only captures the identifiers that relate to the M2M Device/Gateway, not the M2M Network elements (e.g. MAS, MSBF, etc.).

Table 7.1: M2M Device/Gateway Identifiers lifecycle and characteristics

Name	Assigned by	Assigned to	Assigned during	Lifetime	Scope	Provisioned at	Used during	Remarks
Pre-provisioned-ID	Various. (out-of scope)	Various. (out-of scope)	Various. (out-of scope)	Various. (out-of scope)	Various. (out-of scope)	M2M Device/Gateway, MSBF, optionally MAS depending on specific bootstrapping procedure	M2M bootstrapping (clause 8.3)	Several such IDs likely on one M2M Device/Gateway depending e.g. on access type.
M2M-Node-ID	MSBF, unless pre-provisioned	M2M Node	M2M Service Bootstrap, unless pre-provisioned	Depends on deployment	Global	D/G M2M Node, MAS, NSCL	M2M Service Connection procedure (clause 8.4)	M2M Root Key (Kmr), see clause 8.2.2.1, is associated with this ID.
M2M-Connection-ID	Network M2M Node, or MAS depending on specific M2M Service Connection procedure	M2M Service Connection	M2M Service Connection procedure	Equal to M2M-Connection-lifetime. Less than or equal to M2M-Node-ID lifetime	NSCL	D/G/N M2M Node	M2M Service Connection procedure (clause 8.4) mId security procedures (clause 8.5)	Kmc is associated with this ID.
D/GSCL-ID	NSCL+MSBF, or NSCL+MAS unless pre-provisioned	D/GSCL	M2M Service Bootstrap, or first M2M Service Connection procedure unless pre-provisioned	Equal to D/GSCL lifetime	Global	D/GSCL (pre-provision case), NSCL+MSBF (bootstrap), NSCL+MAS (service connection)	Resource related procedures (clause 9.3.2)	In the current specification, there is only one D/GSCL-ID per M2M-Node-ID. Furthermore, M2M-Node-ID and D/GSCL-ID may be set to the same value.
NSCL-ID	Out of scope unless pre-provisioned	NSCL	Out of scope	Equal to NSCL lifetime	Global	Out of scope	Bootstrap, Service connection	

NOTE: The format of the identifiers are specified in [1] except for the Pre-provisioned-ID.
All references to "Out of scope" means out of scope for the current version of the present document.
"A+B" means both A and B functionalities are required.

Figures 7.1 and 7.2 provide an overview of Identifiers provisioning and their relationship to the procedures and security keys as per clause 8 of the present document.

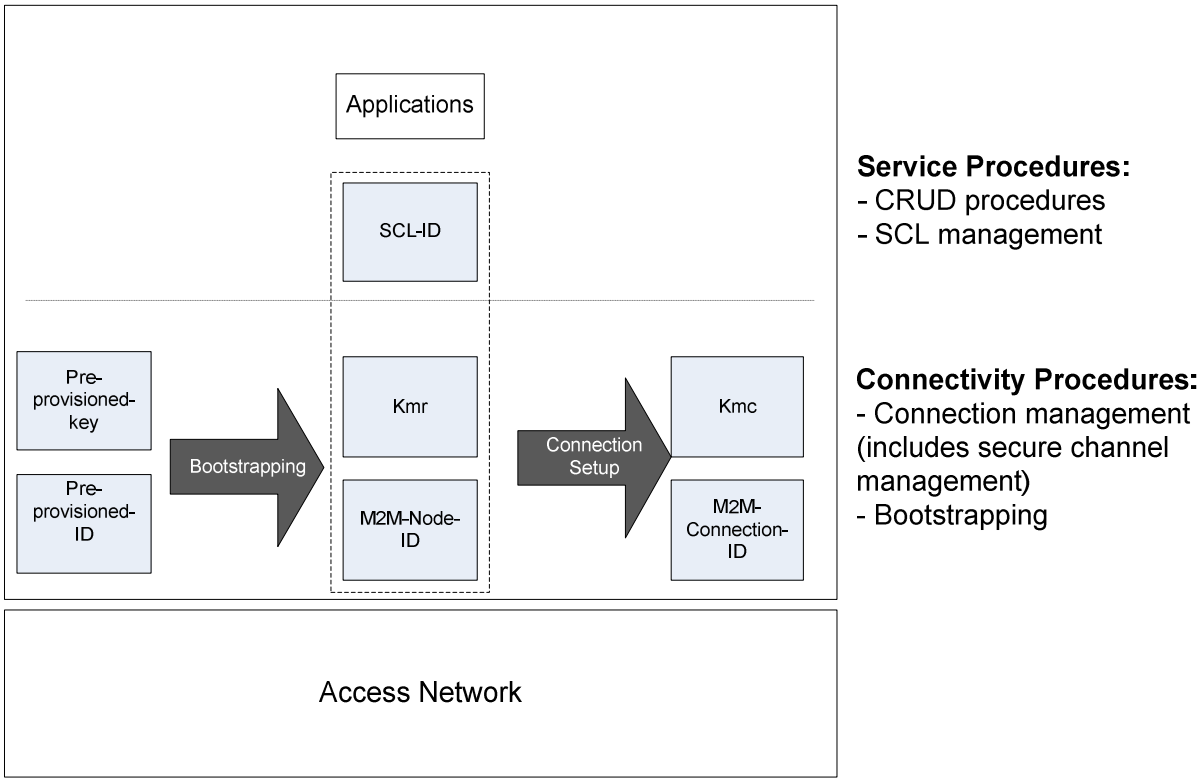


Figure 7.1: M2M identifiers provisioning overview

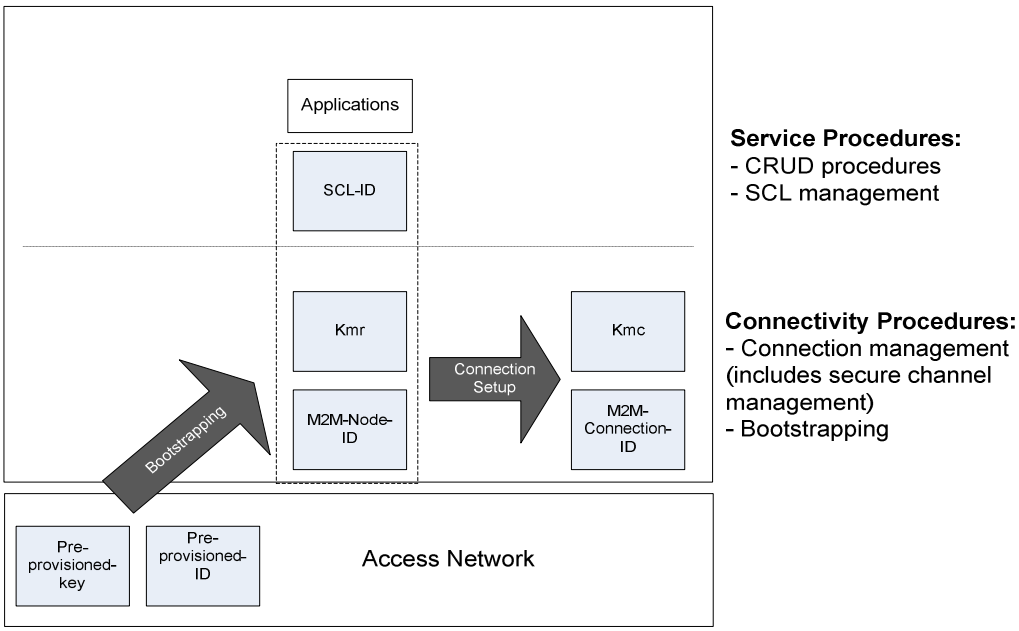


Figure 7.2: M2M identifiers provisioning overview when M2M bootstrap is performed through leveraging access network credentials

7.3 M2M Application Addressing

7.3.1 Introduction

In M2M communication, the goal of M2M addressing is to reach the M2M SCL with which the target M2M application is registered and ultimately the correct application on the M2M Device/Gateway on which the target application is resident. This principle applies to all applications: DA, GA.

Reachability and routing from/to DA and GA M2M applications is always associated with the SCL, with which these applications are registered, and which is resident on M2M Device/Gateway connected to the access network. Reaching an application shall be performed through reaching the SCL the application registered to. An M2MPoC (M2M Point of Contact) shall provide the set of information needed to reach an SCL from a network perspective. Typically an M2MPoC contains information that is e.g. is resolved into a network address.

7.3.2 Application Reachability

7.3.2.1 M2M Communication Point of Contact (M2M PoC)

The M2M PoC shall be used by the M2M system to communicate with a GSCL or DSCL. Once communication with an SCL is achieved, any application registered in the SCL can be reached as long as this application can be uniquely identified.

The M2M PoC for an SCL shall be provided to the M2M system at registration of an SCL. For D' devices, the M2M PoC shall be provided at registration of the GSCL. For D devices, the M2M PoC shall be provided at the registration of the DSCL.

The information included in the M2M PoC, as well as the refresh of the M2M PoC, depends on the access network and the M2M transport device capabilities.

7.3.2.2 Principles guiding Locating Applications

Locating a D/GA M2M application is a two-step process as follows:

- In step 1, there is a need to locate the M2M SCL (DSCL, GSCL) where the M2M application (GA/DA) is registered. Locating the M2M SCL shall be accomplished as follows:
  - For DAs associated with D devices the M2M PoC of the DSCL registered object shall be used for that purpose.
  - For DAs associated with D' devices and GAs, the M2M PoC of the GSCL registered object shall be used for that purpose.
- In step 2, the GSCL and DSCL shall locate the appropriate D/GA M2M application using the registered application Identifier for that purpose.

7.3.2.3 Usage of M2M PoC by the M2M System

The M2M PoC holds the information used by the M2M system to locate routing information for an SCL. This information shall be provided by the registered SCL at registration time. However, locating routing information related to an SCL (and ultimately to the desired application) in an M2M system depends on the characteristic of the access network. This impacts the criteria for updating the M2M PoC by the registered SCL (in addition to the regular SCL registration refresh) and the information to be conveyed in there as well to support the access network specifics.

In general the easiest routing information related to an SCL is achieved when a static public IP address is assigned to M2M Device/Gateway, as it is possible to rely on direct DNS address translation or dynamic DNS address translation.

In those circumstances, the M2M PoC for an M2M registered SCL shall have a URI conforming to RFC 3986 [38] as follows:

URI = scheme:/fullyqualifieddomainname/path/; or

URI=scheme://ip-address/path/.

The following clauses further specify the information to be conveyed in the M2M PoC by a registered SCL for the various access networks, as well as criteria for updating the M2M PoC for the registered M2M SCL, in addition to the normal M2M SCL registration refresh.

7.3.2.3.1 M2M PoC related to M2M SCLs associated with a Fixed Network

In this case the M2M PoC for an M2M registered SCL shall have a URI as described above. If the IP address is private, then the address is usually built based on the address of the related PPP protocol which is a public IP address. This in turn is mapped to the corresponding private address.

7.3.2.3.2 M2M PoC related to M2M SCLs associated with Mobile Networks

If the IP address for the registered SCL cannot be reliably used, and as such cannot be included in the M2M PoC, then in this case the M2M PoC for the registered SCL shall include appropriate information as defined by various access networks.

Each access network shall specify the means for allowing an M2M SP to fetch the IP address associated with an SCL attaching to that access network and consequently the information to be included in the M2M PoC for the registered SCL.

In the event that the M2M SP has connections to multiple access networks, there is a need to establish a binding between the registered SCL and the access network. That binding may be established through SCLs explicitly listing the access network at registration/update time, otherwise the M2M SP may derive it (using the link over which the registration arrived), store it and bind it to the registration information.

7.3.2.3.3 M2M PoC to M2M SCLs associated with multiple access networks

When the M2M transport device attaches to a fixed network, the M2M PoC for a registered M2M SCL shall conform to the procedures associated with a fixed network.

When the M2M transport device attaches to a mobile network, the M2M PoC for a registered M2M SCL shall conform to the procedures associated with mobile networks.

If an M2M transport device already attached to an access network attaches to a new access network, the M2M SCL shall update its reachability data.

8 M2M Security, M2M Service Bootstrap, Service Provisioning and M2M Service Connection procedures

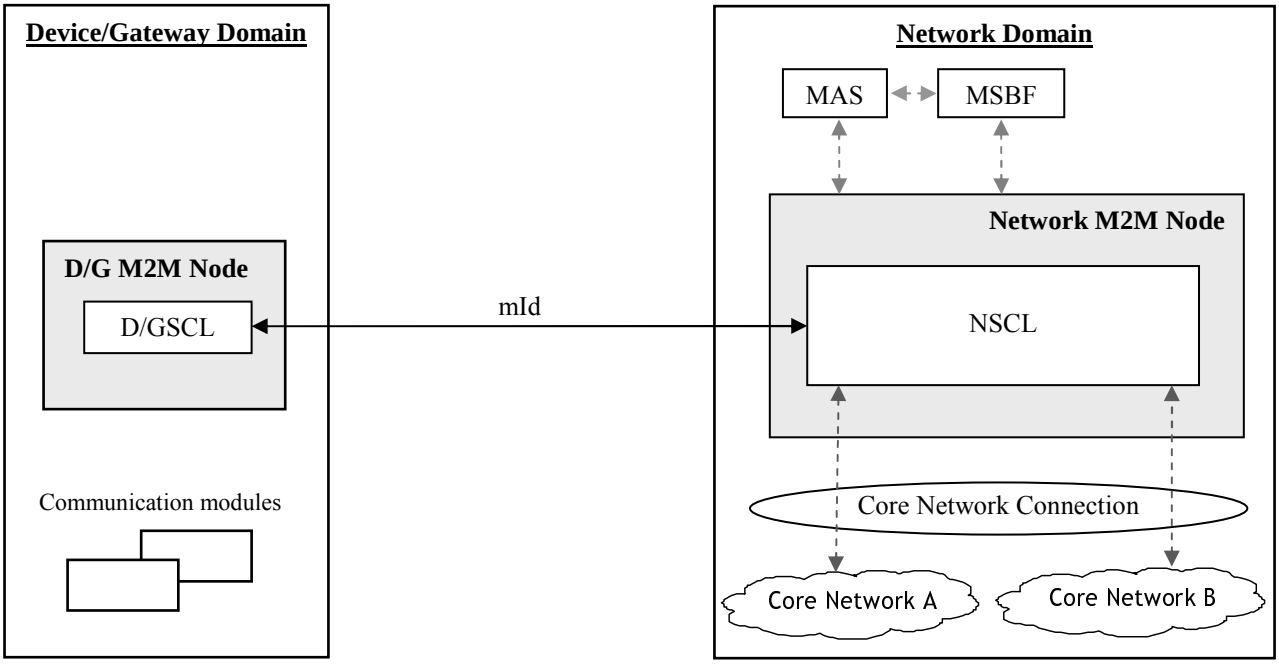
8.1 Introduction

M2M security framework in clause 8.2 lays down the underlying functions and key hierarchy pertaining to M2M security. Clause 8.3 addresses the bootstrapping and service provisioning of D/G M2M Nodes. Clause 8.4 describes the security procedures for M2M Service Connection between the D/G M2M Node and the Network Domain.
Clause 8.5 addresses the security of the mId interface.

8.2 M2M Security Framework

8.2.1 Overview

Figure 8.1 presents the functional architecture from clause 5.1.1 in the context of the security framework.



NOTE: Reference points shown with dashed lines denote interfaces that are out of scope of the present document.

Figure 8.1: Functional architecture elements for bootstrapping

Security framework requirements for Authentication, Key agreement and establishment that enable M2M Service Bootstrap, provisioning and M2M Service Connection procedures are grounded on a clearly defined key hierarchy of the M2M Node. The following clauses define this key hierarchy, authentication, key agreement and authorization.

8.2.2 Key hierarchy and realization

8.2.2.1 Description of M2M keys

The following describes keys used for different levels of Authentication and Authorization in current M2M architecture:

- Kmr - M2M Root Key. This key shall be used for mutual authentication and key agreement between the D/G M2M Node and the M2M Service Provider, i.e. it shall be used by the D/G M2M Node to authenticate the M2M Service Provider, and by the M2M Service Provider for authenticating the D/G M2M Node. Kmr is also used for deriving an M2M Connection Key (Kmc), see below, through authentication and key agreement between the D/G M2M Node and the Network M2M Node. Kmr shall be coupled with a unique D/G M2M Node and M2M Service Provider through an M2M-Node-ID identifier, and may be bootstrapped in the D/G M2M Node in various different ways, depending on the trust relationships between different stakeholders in the M2M ecosystem. At the Network M2M Node side, Kmr shall be stored in a Secured Environment [i.6], within MAS. The Secured Environment shall protect the information within (e.g. Kmr and the Kmc derivation process) from access or manipulation by unauthorized entities. In the M2M Device/Gateway, Kmr shall be stored and used within a Secured Environment Domain controlled by the M2M service Provider. There exists one Kmr for each set of provisioned D/G M2M Node credentials.

  In scenarios where the M2M Service Provider and the Access Network Provider have a trust relationship (including the case that they are actually the same entity), access network credential may be used as Kmr.

Kmc - M2M Connection Key. This key shall be derived from Kmr, upon successful mutual authentication of the D/G M2M Node, as explained above. Upon derivation, Kmc shall be delivered from the MAS (wherein it is derived within the same Secured Environment as Kmr) to the Network M2M Node, where it is stored in a local Secured Environment Domain. Kmc shall expire upon termination of the corresponding M2M Service Connection. Lifetime of Kmc shall be less than or equal to the lifetime of Kmr. A different Kmc shall be generated for every new M2M Service Connection procedure of the D/G M2M Node with the same or a different Network M2M Node. Kmc is used as symmetric shared secret for setting up secure application data sessions between a Network M2M Node and a D/G M2M Node. As an example, if the HTTP protocol is used for data transfer over the mId Reference Point, and if TLS-PSK is used as a method for secure HTTP session establishment on the mId Reference Point, then Kmc may serve as the PSK for TLS. Key material further derived from Kmc during the setup of such security association may be used for securing the data transport over the mId Reference Point, e.g. facilitate operations such as encryption and integrity protection on the data that is transferred between a Network M2M Node and a D/G M2M Node. The secure transport session that is established using Kmc is always terminated at a Network M2M Node and a D/G M2M Node.

Lifetime of Kmr is more than or equal to the lifetime of Kmc. In particular, Kmr shall be bootstrapped into a Secured Environment Domain of the D/G M2M Node as per one of the methods described in clause 8.2.2.2 as well as in clause 8.3 and has the permanent role of being used for mutual authentication of the D/G M2M Node with the M2M Service Provider. Upon derivation, Kmc shall be valid for as long as the M2M Service Connection of the D/G M2M Node is valid; Kmc may expires or gets invalidated based on M2M Service Providers' policies.

For the scenario where the Network Operator and the M2M Service Provider do not share a business relationship, the setup of Kmr shall be the responsibility of the M2M Service Provider (in this case Kmr is not setup in the network). Clause 8.3 provides different bootstrapping procedures pertaining to such a scenario.

8.2.2.2 M2M Root Key Provisioning

Clause 8.4 specifies a variety of options for the M2M Service Connection procedure:

- Some M2M Service Connection procedure options (for example, in clauses 8.4.2 and 8.4.3) require a dedicated M2M Root Key Kmr in a Secured Environment Domain [i.6]. The current clause summarizes the options for getting Kmr onto a Secured Environment Domain of a D/G M2M Node.

- Other M2M Service Connection procedure options (for example, in clause 8.4.4) leverage key material that is derived from the access network credentials, which shall be used as M2M Root Key, Kmr. This approach is applicable when the Access Network Operator and the M2M Service Provider is the same stakeholder.

The M2M Architecture supports multiple scenarios for provisioning the M2M Root Key into a destination Secured Environment Domain.

The Secured Environment Domain protecting M2M Root Key may be part of a secured environment integrated with the M2M Device/Gateway or it may be hosted on an Independent Security Element (ISE) [i.6]. A Secured Environment Domain controlled by a trusted stakeholder may assist in provisioning the M2M Root Key to the intended Secured Environment Domain. Strong security recommends that all secured environment domains that need to exchange information in the M2M Device/Gateway are hosted on a single secured environment, whether it is an Independent Security Element or not. In the event that the Secured Environment Domains are hosted on separate secured Environments, then a secure channel should be used to protect any Sensitive Data.

- If any Secured Environment Domains [i.6] in any of the below scenarios resides on a Universal Integrated Circuit Card (UICC) [28] (a type of ISE) then the UICC-Terminal Secure Channel specified in TS 102 484 [31] should be used as a standardized mean to protect any Sensitive Data [i.6] transferred across the interface from the UICC to other Secured Environments in the M2M Device/Gateway. The method of setting up the UICC-Terminal Secure Channel is not specified in the present document.

- When the Secured Environment Domains are hosted on different Secured Environments not involving a UICC, the way to secure sensitive data between the Secured Environments is not specified in the present document.

The following scenarios are supported by the M2M architecture:

1) The M2M Device/Gateway may be provided with M2M Root Key Kmr inside a Secured Environment Domain [i.6] during manufacture or deployment. In these cases, the M2M Service Provider is responsible for ensuring that M2M Device/Gateways are provided with necessary M2M Root Keys Kmr using mechanisms that are not specified in the present document.

This Secured Environment Domain containing Kmr may be hosted by the same Secured Environment used for Access Network Credentials, e.g. a UICC. This scenario is particularly applicable when the Access Network Operator and the M2M Service Provider are the same stakeholder or share a business relationship.

2) The M2M Device/Gateway may leverage key material that is derived from Access Network Credentials, and use that key material to provision the M2M Root Key Kmr in a Secured Environment Domain on the M2M Device/Gateway. Clause 8.3.2 describes such methods.

3) The M2M Root Key Kmr may be provisioned in a Secured Environment on the M2M Device/Gateway in an access-network independent procedure. This scenario is applicable when the Access Network Operator and the M2M Service Provider do not share a business relationship and/or do not wish to use Access Network Credentials for bootstrapping of M2M service layer credentials. Methods for access-network independent provisioning of root keys are described in clause 8.3.3.

For options 1 and 3 there may be no need for Access Network Credentials, depending on the access network technology used.

Properties of the M2M Root Key shall comply to [1].

8.2.2.3 Secured Environment Domains

Secured Environment Domains refer to logical entities that are securely isolated from each other, whether they are separate Secured Environments or are inside a single Secured Environment. Sensitive Functions [i.6] (including the storage and handling of sensitive data such as credentials and key material) shall be protected inside a Secured Environment Domain controlled by its stakeholder.

The M2M Service Provider owning an M2M Node on an M2M Device/Gateway shall control its own Secured Environment Domain.

Providers of M2M applications may control an independent Secured Environment Domain on an M2M Device/Gateway.

M2M Devices/Gateways may also contain a Trusted Environment and a Secured Environment Domain used for Integrity Validation:

- This Secured Environment Domain shall be provisioned with a key used to sign the integrity validation measurements and the pass/fail results.
- This Secured Environment Domain may use asymmetric cryptography for the provisioning and validation of trusted reference values. The trusted reference values may be provisioned by means of digital certificates.

M2M Devices/Gateways may contain Secured Environment Domains owned by other stakeholders such as Access Network Operators (e.g. 3GPP USIM [45] on UICC [28]). Credentials stored in such Secured Environments may be used in the context of the present document when proper agreements are in place between the stakeholders.

8.2.3 M2M Node Security Functionalities

M2M Security functionalities support the security requirements for the Network, Device and Gateway M2M Nodes.

8.2.3.1 Network M2M Node Security Functionalities

The Network M2M Node performs mutual authentication with D/G M2M Nodes and derives keys that are used to setup secure transport sessions with D/G M2M Nodes.

- The Network M2M Node obtains M2M Connection Keys (Kmc) from the MAS.

For M2M Service Connection the Network M2M Node:

- Performs M2M Service Connection procedures with D/G M2M Nodes based on the corresponding D/G M2M Node procedures defined in the clauses below. M2M Service Connection shall be provided through mutual authentication (between a D/G M2M Node and the M2M Service Provider's MAS) and M2M Connection Key (Kmc) agreement. For such a mutual authentication, a bootstrapped or pre-provisioned M2M Root Key (see clause 8.2.2.2) may be used as a permanent shared secret between the D/G M2M Node and MAS.

- Controls the access of D/G M2M Nodes to the Network M2M Node according to policies defined in the MAS.
- Interfaces with the MAS to obtain material needed to perform mutual authentication and key agreement with D/G M2M Nodes and serves as the "authenticator".
- Establishes secure transport data sessions over the mId Reference Point.

For M2M Device/Gateway Integrity Validation, if supported and required by its local policies, the Network M2M Node:

- Determines whether an M2M Device or M2M Gateway supports Integrity Validation.
- If an M2M Device or M2M Gateway supports Integrity Validation, validate the integrity based on the reported Integrity Validation status and provisioned security policy, and then trigger policy-determined post-validation actions such as:
  - Access control (full, partial or denial).
  - IUpgrade, remediate or roll-back firmware or software of the M2M Device or M2M Gateway.
- Signs any Integrity Validation related policy information communicated to an M2M Device or M2M Gateway using a pre-provisioned key.

8.2.3.2 Device/Gateway M2M Node Common Security Functionalities

The D/G M2M Node performs the following operations within a Secured Environment Domain:

- Obtains an M2M Connection Key (Kmc) and protects the Kmc from unauthorized access.

The D/G M2M Node:

- Initiates M2M Service Connection procedures involving mutual authentication with the M2M Service Provider. For this, the Kmr key that is stored in the Secured Environment Domain of the D/G M2M Node. As a result of successful authentication, an Kmc is derived within the Secured Environment, as specified in clause 8.2.2.1.

For M2M Device/Gateway Integrity Validation, if supported by the M2M Device/Gateway, the D/G M2M Node:

- Performs verification and/or reporting of the integrity status of the M2M Device/Gateway, at all times other than during M2M Service Bootstrap or during any operation in which its integrity has not yet been validated.
- Retrieves the signed integrity validation status records from the Secured Environment and verifies the integrity of the records by means of a provisioned key.
- The integrity validation results may be signed by the Secured Environment Domain for Integrity Validation and communicated to the Network M2M Node.
- Maintains a provisioned security policy for fault-recovery mechanism triggered by receipt of failed Integrity Validation result.
- Policy information for all executable code to be integrity validated is pre-provisioned or is securely downloaded (i.e. included in a signed certificate). Each executable code that requires integrity validation has a trusted reference value and policy information associated with it indicating whether or not Integrity Validation is required for that executable code.
- Optionally (for M2M Device/Gateways that are capable) supports procedures of secure time synchronization.

8.2.3.3 Gateway M2M Node Specific Security Functionalities

For M2M Gateway Integrity Validation, if supported by the M2M Gateway, the Gateway M2M Node:

- According to policies, controls the access of M2M Devices to the Network M2M Node in the case where faults are detected concerning M2M Device integrity.

8.2.4 M2M Device/Gateway Integrity Validation (Optional)

8.2.4.1 Integrity Validation Functional Description

M2M Devices/Gateways may optionally support Integrity Validation (IVal). The requirements for IVal specified herein for M2M Service Layer shall apply whether IVal of M2M Service-Layer functions is either performed as part of a general secure boot process for the M2M Device/Gateway or is performed as part of M2M Service-Layer procedures and consists of:

a) The integrity of executable code is verified by comparing the result of a measurement (typically a cryptographic hash) of the executable code to its trusted reference value. If these values agree, the executable code is successfully verified.

b) The only permitted exception is executable code implemented in physically immutable form, in which case such executable code can be excluded from the IVal procedures.

c) Each executable code shall be individually identified by the TrE and by the Secured Environment for the purpose of IVal.

d) For each of the integrity checks of executable code, the corresponding trusted reference value is obtained from memory. The trusted reference value is protected for authenticity and integrity, otherwise it cannot be used.

NOTE: Protection of trusted reference values may require the M2M Device/Gateway to support asymmetric cryptography and the root certificates of Certification Authorities.

e) The Trusted Environment [i.6] measures the integrity of executable code in the M2M Device/Gateway, as defined by provisioned security policies. The measurements are time-stamped by the Trusted Environment and stored in the Trusted Environment or in a Secured Environment.

f) The Secured Environment [i.6] compares the measurement to the trusted reference value. The executable shall not be allowed to execute if its IVal fails. Execution may, according to provisioned policy, be permitted if the IVal is pending but the M2M Device/Gateway may impose restrictions as to what the executable code is permitted to do.

g) The Secured Environment may be integrated with the Trusted Environment or securely connected to it. In the latter case, keys are provisioned into both environments to enable secure communications. The resulting IVal status records are time-stamped and signed (using a key stored therein for that purpose) by the Secured Environment.

h) The Integrity Validation of an M2M Device/Gateway is successful if all executable code that requires Integrity Validation in the M2M Device/Gateway is successfully verified.

The IVal status of the executable code, (i.e. pass/fail/pending), or simply an alarm/NAK message, shall be communicated to the NSCL, if required by provisioned security policies, and if supported by the context in which IVal is being carried out.

8.2.4.2 Integrity Validation prior to M2M Service Bootstrap (Optional)

Integrity validation (IVal) [i.6] is an optional feature for M2M Devices and M2M Gateways. An Integrity validation procedure is performed prior to executing the M2M Service Bootstrap procedures.

Only components which are stored in physically immutable memory can be exempt from the IVal procedures.

The requirements for IVal specified herein for M2M Service Bootstrap shall apply whether IVal of M2M Service-Layer functions is performed as part of a general secure boot process for the M2M Device/Gateway, or is performed as part of M2M Service-Layer procedures.

The IVal procedure includes the following specific functions:

1) The IVal procedure prior to M2M Service Bootstrap is initiated and controlled by a dedicated function of the M2M Device or Gateway. The dedicated function supports all aspects of the IVal process as defined in [i.6] and as described in the section on general principles of IVal, including the use of Trusted Environment [i.6] and Secured Environment [i.6].

2) The M2M Service Bootstrap procedure is initiated by a dedicated function of the M2M Device/Gateway. This function is logically separate from the dedicated function 1 above. Function 2 is configured to select appropriate steps to be undertaken, according to whether the SCL or Reference Points mId or dIa have failed the IVal test. The authentication and authorization required for M2M Service Bootstrap are not allowed to proceed if the IVal test detects any faults in the SCL or Reference Points mId or dIa.

   An alternative instance of M2M Service Bootstrap functionality may allow M2M Service Bootstrap to proceed if those SCs and Reference Points required for the M2M Service Bootstrap procedure pass the IVal test but other SCs or Reference Points fail the IVal test or have not yet been tested. Complexities introduced by such an instance are not considered in the diagram below but are specified in the section on SCL Management.

3) In the Networks Domain, the protocol end-point for the M2M Service Bootstrap response from the M2M Device/Gateway is optionally provisioned with the corresponding IVal configuration and policy for the M2M Device or Gateway, to allow the alarms of NACK messages to be interpreted and to allow the security attributes of an M2M Device/Gateway to be obtained. If that protocol allows or requires at least the payload of the alarm message to be secured by cryptographic means, the keys provided for that purpose shall be stored in the Secured Environment that is used for IVal and the cryptographic operations shall be performed therein.

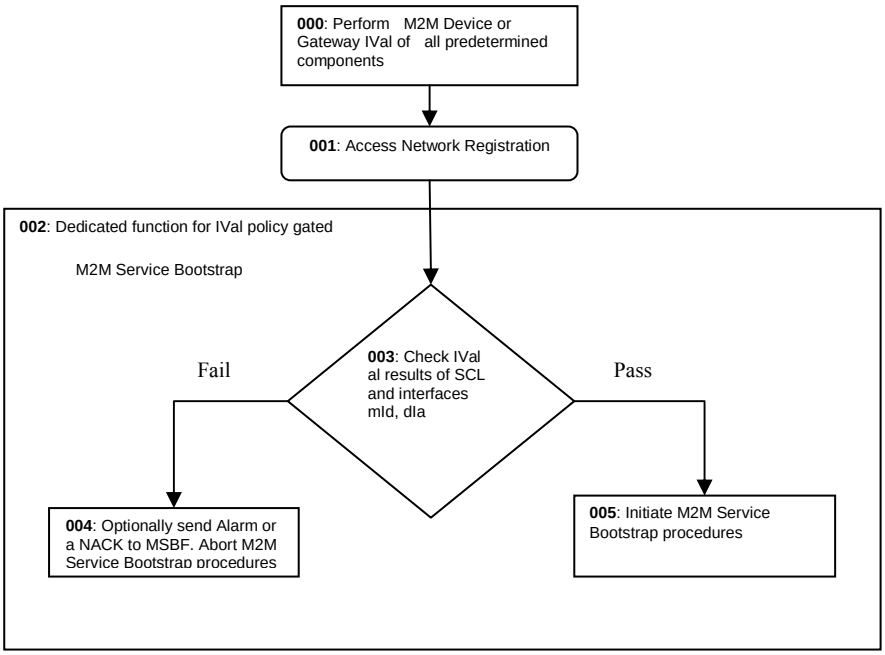


Figure 8.3: IVal as a precondition for M2M Service Bootstrap

Process steps:

Step 000: M2M Device/Gateway performs IVal test of all the SCL and Reference Points mId and dIa, according to function 1 above. It is out of scope to specify exactly when this step occurs, provided it occurs prior to the M2M Service Bootstrap procedure being initiated. For example, the IVal may be performed as part of a secure boot process.

Step 001: M2M Device/Gateway performs access network registration. Details are out of scope.

Step 002: The M2M Device/Gateway activates the dedicated function 2 above, to initiate M2M Service Bootstrap procedures based on preconfigured IVal policy.

Step 003: Function 2 above selects the next step according to the status of IVal test results, i.e. whether or not the SCL and Reference Points mId and dIa have passed IVal test. If PASS, go to step 005.

Step 004: If any SCL components and/or Reference Points have failed IVal, then function 2 above optionally returns an alarm response or a NACK response, if such functionality is supported by the specified M2M Service Bootstrap protocol. The M2M Service Bootstrap procedure is aborted. The M2M Device/Gateway may also be put on a more limited operational state or be powered off.

The alarm/NACK message should be protected for integrity, authenticity, etc., if such functionality is supported by the protocol used for M2M Service Bootstrap.

Step 005: If all SCL components and Reference Points have passed IVal, then M2M Service Bootstrap continues as normal.

8.3 M2M Service Bootstrap procedures

8.3.1 Introduction

The M2M Service Bootstrap procedures are used to provision a secret key called M2M Root Key in the D/G M2M Node and in the M2M Authentication Server (MAS). In addition to provisioning the M2M Root Key, the M2M Service Bootstrap procedures may result in provisioning any combination of the following parameters to the D/G M2M Node:

- An M2M-Node-ID (defined in clause 7.2.1.2)
- An SCL-ID (defined in clause 7.2.1.3)
- A list of one or more NSCL identifiers that the D/G SCL shall use as the next point of contact

Clause 8.3.2 describes M2M Service Bootstrap procedures based on the access network credentials for different situations. Clause 8.3.3 describes three M2M Service Bootstrap procedures that are completely independent from the access network.

8.3.2 Access Network Assisted M2M Service Bootstrap procedures

This procedure shall be invoked in scenarios where the access Network provider and the M2M service provider share a business relationship, while at the same time it is feasible to use access network security credentials for service layer bootstrapping (i.e. the access network involves security operations, and provides an appropriate interface for M2M Service Bootstrap). SIM and AKA-based credentials can be utilized by both GBA and EAP-based procedures as described in the following clauses.

8.3.2.1 GBA based M2M Service Bootstrap procedure

This M2M Service Bootstrap procedure shall be executed for GBA-capable M2M Gateways and M2M Devices In such cases, GBA (Generic Bootstrapping Architecture) shall be supported as specified in [29], [30] and [51].

The high-level GBA-based bootstrap procedure is depicted in Figure 8.4. The Bootstrapping Server Function (BSF) and the HSS (associated with the USIM/ISIM/CSIM/(R-)UIM) belong to the same network operator. The M2M Service Bootstrap Function (MSBF) shall act as a Network Application Function (NAF) and shall support the Zn reference point towards the BSF.

The direct interface between the D/G M2M Node and the MSBF shall comply with the GBA Ua reference point which is defined in [5].

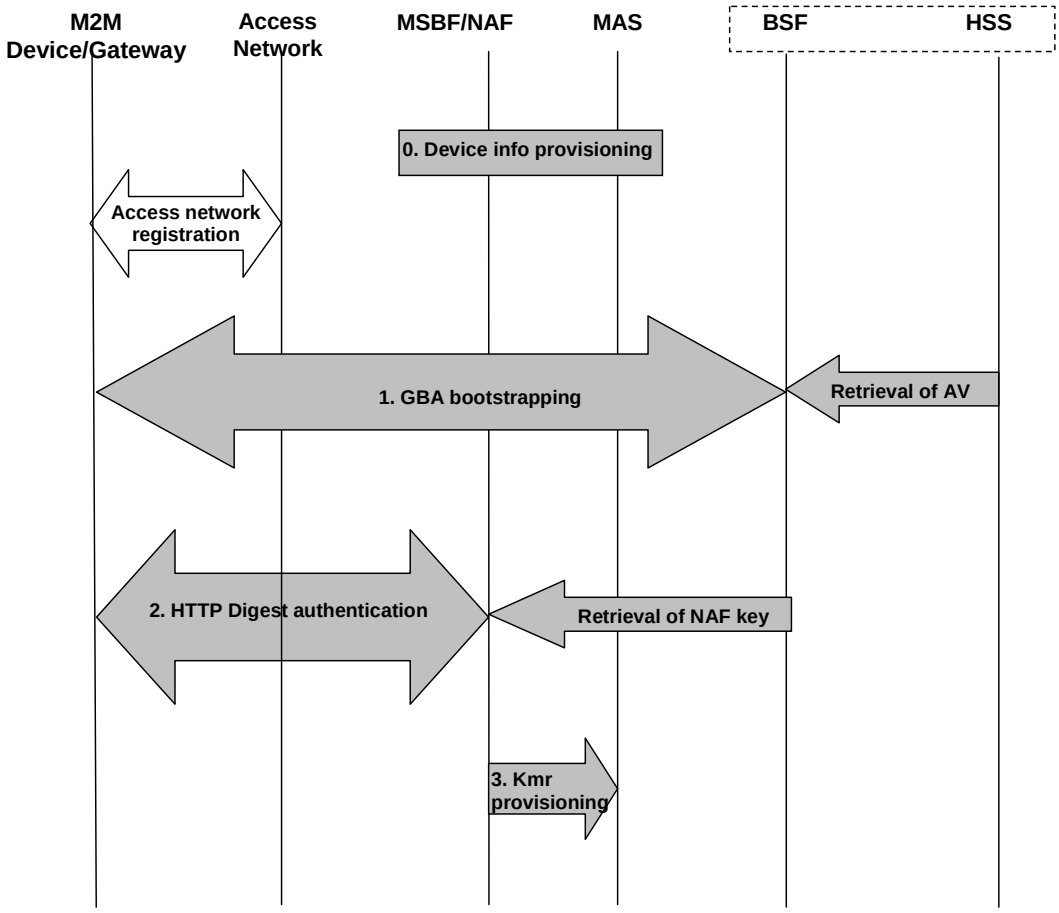


Figure 8.4: M2M Service Bootstrap based on GBA

The high-level procedure is comprised of the following steps:

Step 0: This shall be an offline step. The External Identifiers defined in 3GPP TR 23.888 [i.2] of M2M Devices or M2M Gateways shall be provisioned to the M2M service provider. The External Identifiers can be mapped to the M2M Node-ID and shall also be provisioned in the access network so that it is locally accessible to the BSF.

Step 1: After successful access network registration, the D/G M2M Node shall carry out GBA bootstrapping procedure towards the BSF, using an authentication vector (AV) the BSF fetched from the HSS. The BSF also retrieved the GBA User Security Settings (USS) from the HSS which may contain M2M specific security settings. The details are specified in [29] and annex A of [5] gives signalling flows of the GBA bootstrapping procedure.

Step 2: The D/G M2M Node and the MSBF/NAF shall perform HTTP Digest authentication using the NAF-specific key. The details are specified in [1].

During the HTTP Digest authentication process, the D/G M2M Node sends the GBA bootstrapping transaction identifier (B-TID) obtained from step 1 to the MSBF/NAF. The MSBF/NAF uses the B-TID to retrieve the NAF-specific key External Identifiers and optionally also the M2M specific USS from the BSF, over the Zn reference point as specified in [52]. The MSBF/NAF shall verify that the D/G M2M Node is authorized to access the Network M2M Nonde by checking the retrieved External Identifiers against the pre-provisioned M2M Device/Gateway information (see step 0)..

If the HTTP Digest authentication succeeds, the D/G M2M Node and the MSBF/NAF know that they share the same NAF-specific key. This NAF-specific key is to be used as the M2M Root Key (Kmr). The MSBF/NAF may send a M2M-Node-ID and/or SCL-ID as well as optionally a list of one or more NSCL identifiers to the D/G M2M Node at the end of step 2.

Step 3: The MSBF/NAF shall provision the Kmr, together with the associated M2M-Node-ID, and SCL-ID if provisioned to the D/G M2M Node at step 2, to the MAS.

8.3.2.2 EAP-based Bootstrap Procedure using SIM/AKA-based Credentials

Deployments that want to utilize SIM-based credentials with EAP-based M2M Service Bootstrap procedure shall use EAP-SIM [39] with EAP/PANA. Similarly, EAP-AKA [40] or EAP-AKA' [46] shall be used with EAP/PANA when AKA-based credentials need to be used with an EAP-based procedure. When the EAP-SIM or EAP-AKA or EAP-AKA' credentials are stored in a UICC [27] and shall not be exposed in the M2M Device/Gateway, the UICC EAP framework specified in
TS 102 310 [28] shall be used to avoid exposure of the UICC-based credentials used during the EAP authentication process.

EAP/PANA-based M2M Service Bootstrap procedure is agnostic to the authentication credentials and methods. Whether the credentials are based on access network credentials (e.g. SIM, AKA or AKA') or independent credentials (e.g. IBAKE, TLS) does not make any difference with respect to the EAP/PANA procedure details. Therefore, EAP-SIM and EAP-AKA or EAP-AKA' shall be carried over EAP/PANA according to the general procedure defined in clause 8.3.3.3.

8.3.2.3 Bootstrap Procedure Utilizing EAP-based Network Access Authentication

In this approach the M2M bootstrapping procedure is a by-product of the network access authentication procedure. More specifically, the network access authentication procedure is utilized for the generation of Kmr. Instead of authenticating the M2M Device/Gateway twice (once for the network access, and once for the M2M bootstrapping), it is authenticated once for the network access, and the resultant keys are used for generating the Kmr.

This mechanism is applicable only to the networks using EAP-based mutual authentication and key agreement for network access (e.g. WiFi, Ethernet, WiMAX, Zigbee®, etc.). Furthermore, there shall be a relationship between the network access provider and the M2MSP, so that they can share keying material among themselves. Such relationships include but are not limited to the two being the same operator.

Figure 8.5 depicts the high-level call flow.

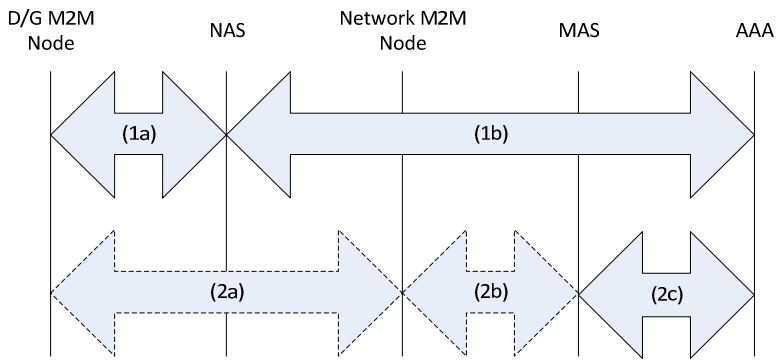


Figure 8.5: M2M Service Bootstrap from network access authentication

In Figure 8.5, NAS represents the Network Access Server, and AAA represents the Authentication, Authorization, Accounting server that are used for mutually authenticating the D/G and the network for the network access service. These are the entities normally used for the network access authentication procedure.

Since the AAA server is responsible for generating the Kmr, it virtually assumes the role of MSBF within the context of M2M architecture.

AAA server and MAS may be co-located.

Step 1: EAP-based network access authentication.

In this step, EAP shall be used for mutually authenticating the D/G and the AAA server to each other via the NAS. Specific EAP method used depends on the access network type and the deployment. EAP is carried over an EAP lower-layer (e.g. IEEE 802.1X [i.4], IEEE 802.16e [i.5] PKMv2, etc.) between the D/G and the NAS, and a AAA protocol between the NAS and the AAA server.

At the end of successful authentication, EMSK (Extended Master Session Key) shall be generated on the D/G and the AAA server.

Step 2: M2M Service Connection procedure triggering the delivery of bootstrapped credential from the AAA server to the MAS.

EMSK shall be used for generating the Kmr on the D/G and the AAA server using the same formula as provided in clause 8.3.3.3.

Kmr = Hash (EMSK, "ETSI M2M Device-Network Root Key" | M2M-Node-ID | M2M-SP-ID)

The required EMSK is already made available to the end-points by the specific EAP method executed.

M2M-Node-ID and M2M-SP-ID parameters shall be either sent explicitly from the NAS to the D/G (this option is applicable only when the EAP lower-layer between the NAS and the D/G is extensible, such as with PANA), or deduced by the D/G by some mechanism not specified in the present document (e.g. M2M-Node-ID is same as Pre-provisioned-ID of the M2M Device/Gateway, and M2M-SP-ID is same as network access provider ID).

Kmr does not have to be generated each time an EMSK is generated. Instead, it can be generated only when it needs to be used. When needed, it shall be generated by using the most fresh EMSK available. More specifically, the D/G can generate the Kmr when it needs to use that key for M2M Service Connection procedure (Step 2a). The AAA server can generate Kmr when the MAS requests that key (Step 2c) in order to perform D/G authentication (Step 2b).

When the MAS and the AAA server are co-located, delivery of credentials from the AAA server to the MAS becomes an internal procedure.

When the MAS enters the M2M Service Connection procedure via the Network M2M Node (Steps 2a and 2b), it shall request the Kmr from the AAA server (Step 2c). AAA server shall generate and deliver the key to the MAS upon receiving such a request.

The MAS and the AAA server shall either use the same identifier for the D/G M2M Node and M2M Device/Gateway respectively, or be aware of the binding between the two identifiers. The interface between the AAA server and the MAS may be implemented using an AAA protocol, such as RADIUS or Diameter.

The MAS shall know the identifier of the AAA server for a given M2M Device/Gateway.

8.3.3 Access Network Independent M2M Service Bootstrap procedures

This procedure is typically applicable in scenarios where M2M Service Bootstrap is not facilitated by the access network. Such scenarios exist when there are no business relationships between access network provider and M2M service provider, or between M2M Device/Gateway manufacturer and M2M service provider, as well as when the access network is inherently unable to cooperate in M2M Service Bootstrap (e.g. in cases where no security operations are configured in the access network layer).

This procedure is also applicable in scenarios such as the one described in clause 8.3.2; i.e. the following access network independent M2M Service Bootstrap mechanism can be employed, even if M2M Service Bootstrap from the access network is possible.

8.3.3.1 M2M Service Bootstrap required properties

The automated M2M Service Bootstrap mechanism, which is performed independently of any access network security operations, has the following M2M architecture-specific properties:

- It does not require the existence of any security infrastructure across network boundaries that constantly issue or perform management of security credentials.

- It is aligned with the M2M architecture, where each D/G M2M Node establishes a secure service session with the M2M service capabilities, and not with other D/G M2M Nodes.
- It is scalable and requires minor signalling, given the extremely high number of M2M Nodes in M2M Devices/Gateways that are potentially deployed under the authority of the same M2M service provider. In addition, it does not require manual provisioning of such keys into servers during M2M Device/Gateway deployment.
- It ensures that the D/G M2M Node and the M2M Service Bootstrap server mutually authenticate each other during the bootstrap procedure, prevents any intermediate server (or other entity) that enables bootstrapping between the D/G M2M Node and M2M Service Bootstrap server to obtain access to the M2M Root Key, and provides perfect forwards and backwards secrecy.
- In cases where the D/G M2M Node switches to a new M2M service provider, it prevents the new operator from obtaining the old M2M Root Key, and enables the new operator to bootstrap a new M2M Root Key.

8.3.3.2 M2M Service Bootstrap Authentication and Transport Options

Either EAP (RFC 3748 [37]) over PANA (RFC 5191 [48]) or TLS over TCP (RFC 4346 [41]) is used for carrying the automated bootstrapping authentication methods supported by the present document. More specifically:

- EAP-IBAKE ([3]) over EAP / PANA supports the IBAKE-based authentication method (clause 8.3.3.3.1).
- EAP-TLS (RFC 2716 [35]) over EAP / PANA supports the Device-Certificate-based authentication method (clause 8.3.3.3.2).
- TLS over TCP supports the Device-Certificate-based authentication method (clause 8.3.3.4).

The procedures when using EAP over PANA for bootstrapping transport are specified below. The procedures when using TLS over TCP for bootstrapping transport are specified in clause 8.3.3.4).

8.3.3.3 Description of EAP over PANA as the M2M Service Bootstrap Transport

At the EAP layer, the D/G M2M Node shall implement EAP peer functionality, and the MSBF shall implement the EAP authenticator functionality.

PANA (RFC 5191 [48]) is used for transporting EAP and bootstrapping-related attributes between the D/G M2M Node and the Network M2M Node Use of PANA for bootstrapping is independent of the "network access authentication", that may or may not be present, and that may use any protocol when present. At the PANA layer, the D/G M2M Node shall implement PANA Client (PaC) functionality, and the MSBF shall implement the PANA Authentication Agent (PAA) functionality.

A AAA protocol (e.g. RADIUS, Diameter, etc.) may be used for transporting EAP and bootstrapping-related attributes between the Network M2M Node and the MSBF. When an AAA protocol is used, the Network M2M Node shall implement AAA client, and the MSBF shall implement AAA server functionality.

Additional network entities may be involved depending on the authentication method being used (e.g. an OCSP server, or a AAA server). The protocol needed between the Network M2M Node and such additional entities are independent of PANA (e.g. OCSP, RADIUS, Diameter, etc.).

The M2M Service Bootstrap procedure flow is depicted in Figure 8.6.

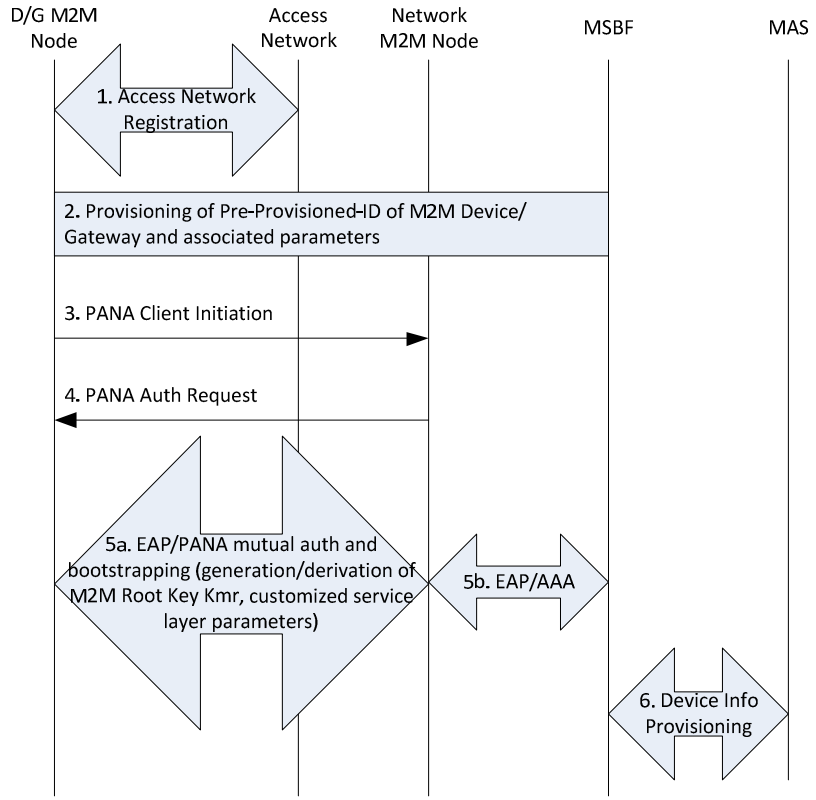


Figure 8.6: M2M Service Bootstrap using automated method

The high-level procedure is comprised of the following steps:

Step 1: Access network registration takes place. This step is independent of the subsequent steps.

Step 2: The Pre-provisioned-ID of the M2M Device /Gateway, as well as a set of other potentially required parameters are provisioned to an MSBF, and also pre-provisioned to the M2M Device/Gateway.

Step 3: The D/G M2M Node sends an M2M Service Bootstrap request (PANA-Client-Initiation message carrying specific AVPs for bootstrapping) to the Network M2M Node. The request contains the Pre-provisioned-ID for the M2M Device/Gateway, along with other potentially required parameters.

This step is optional. M2M Service Bootstrap procedure may also be initiated by the network, in which case Step 3 is omitted.

Step 4: The Network M2M Node receives the M2M Service Bootstrap request, acknowledges the request and provides a set of parameters relevant to the subsequent authenticated key exchange (such as choice of cryptographic functions to be used, etc.) encoded as AVPs in a PANA-Authentication-Request message.

In case the bootstrapping procedure is initiated by the network (i.e. when Step 3 is omitted), this message becomes the first one to be sent between the network and the D/G M2M Node. If the M2M Device/Gateway IP address is known, this message is sent to that address (i.e. it is unicasted). Otherwise, the message is either multicasted or broadcasted to a network where target M2M Device/Gateway is expected to be attached. This message may contain one or more M2M Service Provider identifiers, NSCL identifier (SCL-ID), MSBF identifier, and the Pre-provisioned-ID of the M2M Device/Gateway in the MSBF or MAS. A D/G M2M Node receiving this message shall discard it if the received Pre-provisioned-ID in the message does not match the receiving M2M Device/Gateway's own Pre-provisioned-ID that is pre-provisioned in the M2M Device/Gateway.

Step 5: The D/G M2M Node, the Network M2M Node, and MSBF use the negotiated parameters and exchange a set of transactions in order to mutually derive/deliver an M2M Root Key (Kmr). Mutual authentication is performed by using a negotiated EAP method carried over EAP/PANA between the D/G M2M Node and the Network M2M Node, and over an AAA protocol between the Network M2M Node and the MSBF. Number of messaging and the carried payloads depend on the specific EAP method being used (as defined by EAP-TLS, EAP-IBAKE, etc.). The MSBF may optionally provide a list of one or more NSCL-URIs that the D/G SCL in the D/G M2M Node shall use as the next point of contact.

Note that the Steps 3 and 4 are in fact part of the overall EAP/PANA procedure along with the Step 5a. The former two steps are shown separately in order to highlight their aforementioned functionalities.

The MSBF informs the D/G M2M Node that bootstrapping is complete, along with customized parameters required for the subsequent service session establishment(s). PANA-Authentication-Request with C-bit (Complete) set to 1 indicates the end of procedure. This message may include the M2M-Node-ID assigned by the network to the D/G M2M Node (M2M network can choose to assign its own identifiers that are different than the Pre-provisioned-IDs on the M2M Devices/Gateways). D/G M2M Node acknowledges this by responding with a PANA-Authentication-Answer with C-bit (Complete) set to 1.

The EAP method carried over PANA produces EMSK (Extended Master Session Key, see RFC 3748 [37]) on both the D/G M2M Node and the MSBF. Kmr is derived from that key using the following formula:

Kmr = Hash(EMSK, "ETSI M2M Device-Network Root Key" | M2M-Node-ID | M2M-SP-ID)

Hash is a one-way keyed hash function specified in TS 102 921 [1]. If an M2M-Node-ID is not assigned by the network, then the Pre-provisioned-ID is used as the M2M-Node-ID.

Step 6: The mutually computed M2M Root Key is safely stored within a Secured Environment in the D/G M2M Node. The MSBF provisions the identity of the D/G M2M Node (M2M-Node-ID) as well as the generated M2M Root Key (Kmr) to the MAS. The interface between the MSBF and the MAS are out-of scope of the present document.

8.3.3.3.1 EAP-IBAKE over PANA

This mechanism establishes an M2M Root Key between a type D/G M2M Node and MAS. An M2M Service Bootstrap Function (MSBF) is negotiating parameters with the D/G M2M Node and is facilitating the bootstrap protocol, which is based on Identity Based Authenticated Key Exchange (IBAKE) [2], [3], [4]. The M2M Service Bootstrap procedure assumes that the D/G M2M Node shall be bootstrapped through direct communication with Network M2M Node only; i.e. it is assumed that the D/G M2M Node shall perform service layer bootstrapping using the mId Reference Point.

The procedure leverages the concept of Identity Based Encryption (IBE) [2]. Specifically, a publicly known identity for every M2M Device/Gateway (e.g. MAC Address, etc) shall be temporarily used to derive its IBE public key. An IBE private key for this identity shall be specially generated and associated with this public key by a KGF (Key Generation Function) that is operated/owned by the M2M Service Provider, or by a third party that has trust relationships with the M2M Service Provider. The private key may be provisioned to D/G M2M Node by Network M2M Node i.e. through the mId Reference Point in a secure manner, or may be pre-provisioned by the manufacturer. Similarly, KGF shall generate an IBE private key for MSBF, associated with the IBE public key (identity) of MSBF, upon request. When IBAKE is used for M2M Service Bootstrap, D/G M2M Node and MSBF shall use their IBE private/public keys as per the IBAKE protocol [3] in order to securely derive the M2M Root Key (Kmr). IBE public/private key pairs shall be generated based on date information and hence they expire automatically, upon expiration of the embedded date.

The functional architecture that is considered by this automated bootstrapping method is depicted in Figure 8.1.

8.3.3.3.1.1 High level description of the procedure for D/G M2M Nodes

The high-level M2M Service Bootstrap procedure flow is depicted in Figure 8.7. In this figure, messages involving the D/G M2M Node are shown end-to-end between the D/G M2M Node and the corresponding entity (i.e. MSBF, Network M2M Node); all such messages shall use the mId Reference Point on the D/G M2M Node, i.e. they shall leverage Network M2M Node.

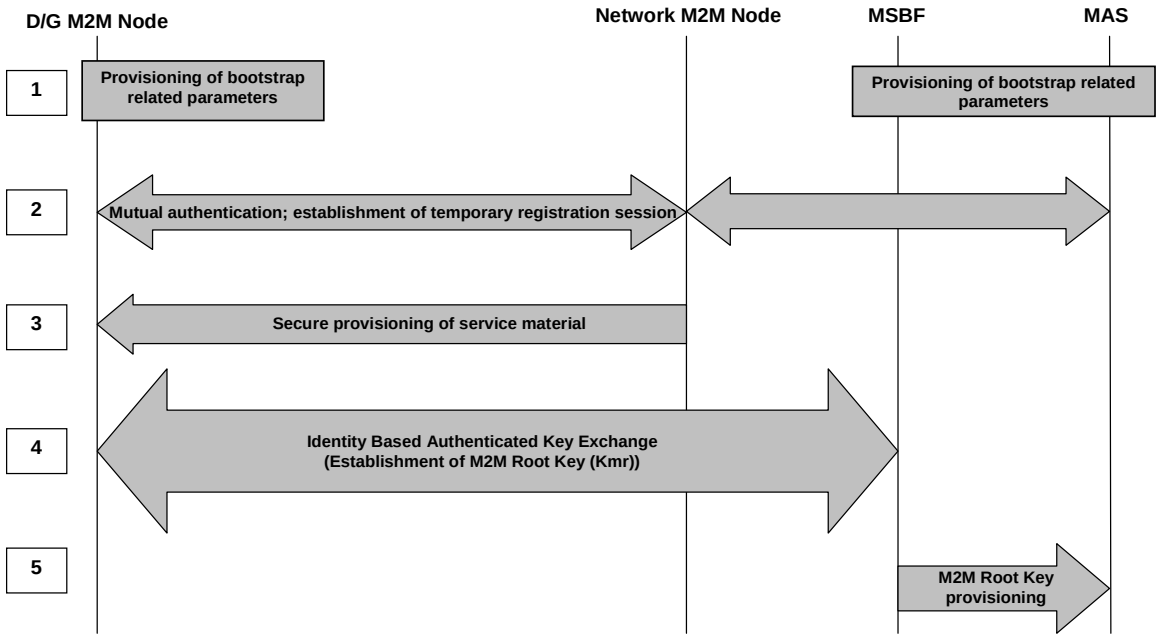


Figure 8.7: M2M Service Bootstrap using IBAKE over EAP over PANA

The high-level procedure is comprised of the following steps:

Step 1: This is an offline step. A Pre-provisioned-ID of the M2M Device/Gateway along with a pre-provisioned secret (such as a key or password, pre-provisioned potentially by the manufacturer), shall be securely provisioned to MAS. In addition, MSBF shall be provisioned with the M2M Device/Gateway Pre-provisioned-ID as well as a set of bootstrap-related parameters, described later. The identity and pre-shared secret pair shall be securely provisioned to MAS and MSBF using methods that are outside the scope of the present document. (Throughout the rest of this clause, any reference to the term "password" also applies to the case where the pre-provisioned/pre-shared secret is a key).

Step 2: The D/G M2M Node shall perform temporary M2M Service Connection with the M2M Service Provider, through Network M2M Node. Using the pre-provisioned secret as a temporary M2M Root Key (which is known to MAS through step 1), the D/G M2M Node shall be mutually authenticated with the M2M Service Provider (MAS shall communicate with Network M2M Node for this, as per the regular mutual authentication procedure). This is required in order for the D/G M2M Node to be authorized to communicate only with MSBF, i.e. for prohibiting unauthorized D/G M2M Nodes to reach MSBF (following the default M2M service provider's authorization procedures, which are also applied during regular M2M Service Connection procedures; see clause 8.4).

Step 3: Through this secure temporary M2M Service Connection, the D/G M2M Node may be provisioned with service material, including material that shall be used for securely bootstrapping a permanent M2M Root Key (see clauses 8.2.2.1 and 8.2.2.2). When needed (i.e. either when the D/G M2M Node is not pre-provisioned with this material or when the M2M Service Provider wishes to provision the D/G M2M Node with new material), such service material shall be sent to the D/G M2M Node in a secure fashion over mId, using key(s) derived during step 3 above, i.e. upon successful mutual authentication of the D/G M2M Node with the M2M Service Provider (see clause 8.2.2.1 where the pre-provisioned secret serves as a temporary M2M Root Key). The NREM service capability may be used for provisioning of such material to the D/G M2M Node (e.g. using OMA DM management objects). Alternative methods, such as PANA, may also be used for securely provisioning such material to the D/G M2M Node.

Step 4: MSBF and D/G M2M Node perform Identity-Based Authenticated Key Exchange. This procedure is further comprised of 3 steps, as explained in [3] as well as in clause 8.3.3.3.1.2. During this procedure, the D/G M2M Node and MSBF perform cryptographic operations in order to mutually calculate an M2M Root Key (Kmr). Upon successful calculation of Kmr, MSBF may provision a value for the M2M Node ID. The MSBF may optionally provide a list of one or more NSCL Identifiers that the D/G SCL in the D/G M2M Node shall use as the next point of contact.

Step 5: The mutually (securely) computed M2M Root Key shall be safely stored within a Secured Environment in the M2M Device/Gateway. MSBF shall provision the generated M2M Root Key (Kmr) and optionally the M2M Node ID to MAS, where it shall be stored within a Secured Environment.

8.3.3.3.1.2 Detailed description of the procedure

8.3.3.3.1.2.1 Provisioning of service material prior to bootstrapping (step 1)

Pre-provisioned-ID and pre-shared secret pair: The M2M service provider shall initially identify the M2M Device/Gateway using a Pre-provisioned-ID that has been pre-provisioned by the M2M Device/Gateway manufacturer. In addition to this identity, an associated pre-shared secret (key or password) shall also be pre-provisioned by the manufacturer, when the IBAKE-based automated bootstrapping procedure is to be applied. The same identity and secret pair shall be securely provisioned to MAS. This can take place e.g. through a secure web interface, or by other secure means, which are beyond the scope of the present document. This pre-provisioned identity/secret pair shall be used for temporary M2M Service Connection of the D/G M2M Node with the M2M service operator's infrastructure. Upon successful mutual authentication (treating this pre-shared secret as a temporary M2M Root Key), the D/G M2M Node shall be authorized to reach MSBF (through Network M2M Node) for mutually agreeing on an M2M Root Key. D/G M2M Node identity and thereby its IBE public key may be provisioned offline to MSBF as well; the provisioning method is out of scope.

IBE private/public keys and parameters: Automated bootstrapping using IBAKE shall be performed between D/G M2M Node and MSBF, as per the IBAKE protocol [3], [4]. For this, the D/G M2M Node shall be provisioned with an IBE private key and an IBE public key (identity). Such material may be pre-provisioned by the manufacturer, or may be securely provisioned online, upon temporary M2M Service Connection (step 4). The IBE private/public key pair shall be generated by a KGF (Key Generation Function), which may either be owned by the M2M service provider, or by a third party that is trusted by the M2M service provider. When KGF is owned/used by the M2M service provider, it may be realized as an M2M NA, and may use the mIa Reference Point. The approach that is followed for provisioning the IBE private key into the D/G M2M Node is use-case dependent and guides decisions regarding expiration of the IBE private key. In particular, if the IBE private/public key pair is pre-provisioned by the manufacturer, then private key expiration should take into account the potential period of time that the D/G M2M Node will not be bootstrapped with the M2M service provider, as well as the properties of the M2M Device/Gateway deployment method/environment. If the IBE private/public key pair is provisioned by NREM, then IBE private keys could expire within a very limited period of time (use-case as well as policy dependent). An IBE private/public key pair shall also be assigned to MSBF offline; the IBE public key of MSBF may be pre-provisioned into the D/G M2M Node by the manufacturer, or provisioned during step 4.

8.3.3.3.1.2.2 Temporary M2M Service Connection and provisioning of service parameters (steps 2, 3)

Given that the D/G M2M Node is aware of the intended M2M service provider, the D/G M2M Node shall perform temporary M2M Service Connection with the M2M service provider in step 2, using the pre-provisioned pre-shared identity and secret that have been pre-provisioned during offline step 1. This step is required in order for the D/G M2M Node to gain connectivity to the Network M2M Node and be authorized to communicate with MSBF through the Network M2M Node.

This step simply re-uses the default M2M Service Connection operations. Mutual authentication shall take place using the pre-provisioned shared secret (key/password), which shall be treated as a temporary M2M Root Key (and which shall be provisioned to MAS along with the Pre-provisioned-ID during offline step 1). This process makes sure that only authorized D/G M2M Nodes shall communicate with MSBF.

As a result of this temporary M2M Service Connection, an M2M Connection Key (Kmc) shall be mutually agreed (see clause 8.2.2.1) and may be used for temporarily securing the mId Reference Point, until bootstrapping is complete.

Note that a "null" password may be alternatively used, in scenarios where bootstrapping is performed within a controlled environment. For example, if the communication between D/G M2M Node and Network M2M Node is already secured using access network security mechanisms (e.g. cases where access network operator and M2M service provider share a limited business relationship), then temporary mutual authentication at the M2M service layer may use a null string as a password (temporary M2M Root Key), just for the purpose of authorizing the D/G M2M Node to reach MSBF. Within such a controlled environment, the already applied inter-network security mechanisms are sufficient to protect delivery of the IBE private key over mId to D/G M2M Node.

Upon successful temporary M2M Service Connection, a set of service related material may be provisioned by Network M2M Node to the D/G M2M Node (step 3), including an IBE private key, the IBE public key of MSBF as well as the corresponding public parameters of the KGF. Secure provisioning of this material shall leverage the security association that was established over the mId Reference Point, upon successful D/G M2M Node authentication. Such provisioning may be omitted, if this material has already been pre-provisioned by the manufacturer (this is known during step 1 and is use-case dependent).

At this point, D/G M2M Node and MSBF have been provisioned with each other's IBE identities, as well as with their IBE private keys, and they are aware of the public parameters of the KGF(s) that are used for generating the private/public key pairs.

8.3.3.3.1.2.3 Bootstrapping of M2M Root Key (Kmr) (steps 4, 5)

The Identity Based Authenticated Key Exchange (IBAKE) procedure for M2M Service Bootstrap of the M2M Root Key shall be comprised of the following messages (included in step 4). Protocol details can be found in [3] and [4].

1) MSBF shall send (through Network M2M Node) an M2M Root Key Establishment Request to D/G M2M Node, along with its IBE public key, as well as other parameters [3]. This message shall be IBE encrypted with the public key of M2M Device/Gateway.

2) D/G M2M Node shall decrypt the message with its private key, and upon successful decryption shall accept the request by responding with an M2M Root Key Establishment Response, which shall include IBE related parameters (public key of M2M Device/Gateway, as well as other parameters specified in [3]). This message shall be IBE encrypted with the public key of MSBF.

3) MSBF shall decrypt the response from D/G M2M Node using its IBE private key; it shall subsequently send an M2M Root Key Establishment Complete to D/G M2M Node, which shall be IBE encrypted with the latter's public key.

The D/G M2M Node and MSBF shall mutually calculate the common permanent M2M Root Key, using publicly available, agreed cryptographic parameters and functions [3], [4]. Upon successful calculation of Kmr, the MSBF may provision a value for the M2M-Node-ID to the D/G M2M Node to be associated with the Kmr. Upon completion of the above process, MSBF shall securely provide the agreed M2M Root Key to the MAS (step 5) using a proprietary interface and shall subsequently delete this key from its local storage. If MSBF provisions a value for M2M-Node-ID to the D/G M2M Node, then MSBF shall securely provide the same M2M-Node-ID to MAS along with Kmr. Hence, as a result of this procedure, an M2M Root Key (Kmr) shall be bootstrapped to D/G M2M Node and MAS, and shall further be used during M2M Service Connection procedure, for D/G M2M Node mutual authentication with the M2M Service Provider, as well as for derivation of Kmc keys, as per the ETSI M2M key hierarchy. The M2M Root Key (Kmr) shall be generated during IBAKE within the Secured Environment of the D/G M2M Node and in MSBF, and shall only be provisioned to MAS by MSBF in a secure proprietary manner; SCL shall never become aware of the value of Kmr.

The MSBF may optionally provide a list of one or more NSCL Identifiers that the D/G SCL in the D/G M2M Node shall use as the next point of contact.

8.3.3.3.1.3 IBAKE over EAP/PANA

Temporary M2M Service Connection as per step 2 above, may take place using EAP over PANA as per clause 8.4.2 for mutual authentication and key agreement using EAP over PANA. Moreover, IBE parameters including IBE private key may be securely sent to D/G M2M Node using the very last PANA message (step 3), upon successful mutual authentication and key agreement (in step 2).

Phase-2 of IBAKE based bootstrapping is also carried over EAP over PANA. Specifically, EAP-IBAKE method [3] for bootstrapping is carried over EAP over PANA in this phase (EAP/PANA may be used over mId, while EAP/AAA may be used between Network M2M Node and MSBF). Kmr is generated at this step in accordance with the formula provided in clause 8.3.3.3.

8.3.3.3.2 EAP-TLS over EAP/PANA

8.3.3.3.2.1 Introduction

The D/G M2M Node and the MSBF perform a mutually authenticated EAP-TLS handshake [42] using Device Certificates and Server Certificates. EAP-TLS runs on top of EAP [37] using PANA [48] for transport.

Using EAP-TLS means letting the EAP negotiate and carry that specific EAP method at Step 5a/5b of the call flow depicted in Figure 8.6.

The procedure of M2M Service Bootstrap, independent of access network is performed by the mutual authentication using TLS handshaking based on EAP-TLS over PANA between the D/G M2M Node and the MSBF. Through the procedure of TLS handshaking, the M2M Root Key is generated from the EMSK which is in turn generated from the TLS session key and then mutually shared between the D/G M2M Node and the MSBF. Finally, the MSBF provisions the MAS with the root key.

8.3.3.3.2.2 High level description of the procedure

The Certificate based architecture is described in clause 8.3.3.5.3.

Figure 8.8 shows the high level flow of the procedure of EAP-TLS over PANA.

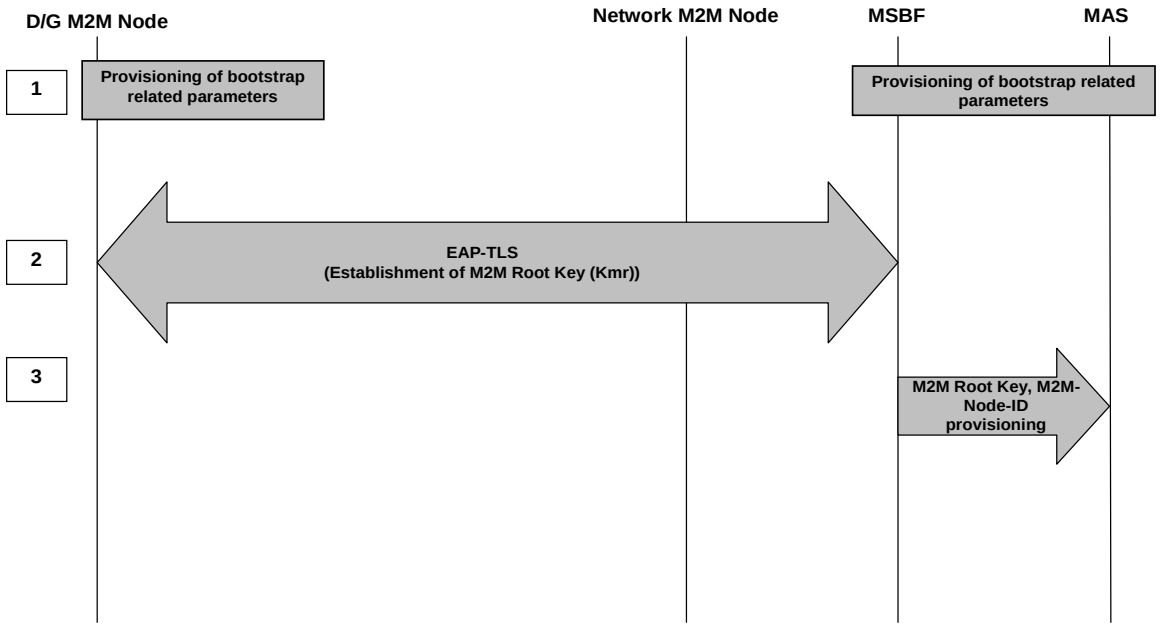


Figure 8.8: M2M Service Bootstrap using EAP-TLS over PANA

The high level procedure is composed of the following steps:

Step 1: The M2M Device/Gateway, the MSBF and the MAS shall be pre-provisioned with the information for M2M Service Bootstrap, including each identifiers, certificates and private/public keys, etc. for them (see clause 8.3.3.5.2 Bootstrap credentials when using device certificates).

Step 2: Through mutual authentication using procedure of TLS handshaking based on EAP over PANA, the D/G M2M Node and the MSBF shall share the same M2M Root Key mutually. The D/G M2M Node shall apply a MSBF Certificate Status Verification Method (see clause 8.3.3.5.4).

Step 3: M2M Root Key and M2M-Node-ID are provisioned from MSBF to MAS.

8.3.3.4 TLS over TCP

Summary: This method uses TLS/TCP with device certificates and server certificates for mutual authentication of the D/G M2M Node and the MSBF. Once a mutually authenticated secure connection is established, then the MSBF remotely provisions M2M-Node-ID and Kmr to a secured environment on the M2M Devices/Gateways. All sensitive computations (on the M2M Device/Gateway side) shall take place in a secured environment. The formatting of messages sent between the MSBF and D/G M2M Node (above the TLS layer) are specified in [1]. The TLS ciphersuite shall conform to [1].

The bootstrapping credentials shall be pre-provisioned onto the M2M Device/Gateway are described in clause 8.3.3.5.2.

Architecture: This procedure uses the architecture described in clause 8.3.3.5.3.1 with the following clarification.

For this procedure the Network M2M Node shall act as a Network Address Translator (NAT [34]) on IP packets containing TLS layer messages received from the D/G M2M Node and MSBF, to direct those IP packets to the correct destination. The D/G M2M Node shall interface to the MSBF (via network address translation by the Network M2M Node) to establish a mutually authenticated and end-to-end secured channel, which shall then be used for transporting Kmr from the MSBF to the D/G M2M Node. Through this interface (between D/G M2M Node and MSBF) any combination of the following parameters may optionally be provisioned:

- An M2M-Node-ID (defined in clause 7.2.1.2).
- An SCL-ID (defined in clause 7.2.1.3).
- A list of one or more NSCL identifiers that the D/G SCL in the D/G M2M Node shall use as the next point of contact.

8.3.3.4.1 Detailed procedures

Preconditions:

- To support this procedure, the Network M2M Node shall have a dedicated IP address for this procedure (that is, Access Network Independent M2M Service Bootstrap using TLS over TCP). This dedicated IP address enables the Network M2M Node to identify packets intended to be used for this procedure, so the Network M2M Node can act accordingly.
  - The Network M2M Node shall perform network address translation to IP packet containing TLS layer messages received from the D/G M2M Node and MSBF, to direct those IP packets to the correct destination. The Network M2M Node shall not perform any processing on the TLS layer messages. That is, the Network M2M Node shall act as a NAT [34].

The M2M SP shall ensure that the D/G M2M Node is directed to the correct IP address for this procedure. Further details are found in [1].

Call Flow.

Figure 8.9 shows a summary of the call flow when using this procedure.

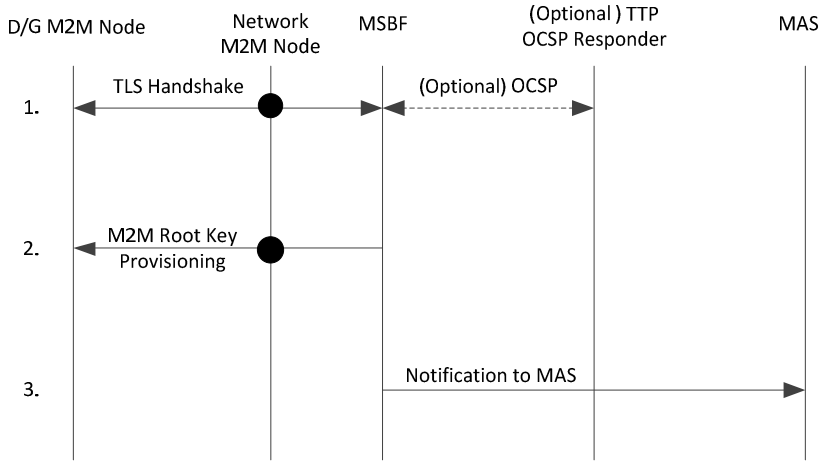


Figure 8.9: Call flow for Automated Bootstrapping using Device-Certificate TLS over TCP

The solid dots on messages exchanged through the Network M2M Node indicate that the Network M2M Node acts as Network Address Translator (NAT [34]) on these messages.

The above procedure is comprised of the following steps:

Step 1: The D/G M2M Node and MSBF shall perform a mutually authenticated TLS/TCP handshake [41], [43] using Device certificates and Server certificates, communicating via a Network M2M Node. The D/G M2M Node shall send its TLS layer messages to a destination IP address at the Network M2M Node that the Network M2M Node has dedicated for "Access Network Independent M2M Service Bootstrap using TLS over TCP". The Network M2M Node shall not process the contents of TCP packets using this reserved IP address (the contents are TLS layer messages), but simply performs network address translation to direct these packets to and from the MSBF. That is, the Network M2M Node shall act as a NAT [34].

- The D/G M2M Node shall apply one of the methods for MSBF certificate status verification (see clause 8.3.3.5.4).
- The MSBF may independently request verification of the status of the M2M Device/Gateway certificate from a TTP - the details are outside the scope of the present document.

If the TLS Handshake is successful, then this results in the D/G M2M Node and MSBF establishing mutually authenticated session keys.

Step 2: The MSBF shall provision the M2M Root Key (Kmr) to the D/G M2M Node, secured at the TLS layer using the negotiated ciphersuite. The MSBF may optionally provide any combination of the following parameters:

- An M2M-Node-ID.
- An SCL-ID.
- A list of one or more NSCL identifiers that the D/G SCL of the D/G M2M Node shall use as the next point of contact.

Step 3: The MSBF shall provision the following parameters to the MAS: the M2M-Node-ID; M2M Root Key (Kmr); and (if known to the MSBF) SCL-ID.

8.3.3.5 Common Aspects of TLS/Certificates-Based M2M Service Bootstrap procedures

8.3.3.5.1 Overview

There are two M2M Service Bootstrap procedures utilizing TLS: one based on EAP-TLS transported using EAP over PANA (clause 8.3.3.3.2), and another procedure based on TLS over TCP (clause 8.3.3.4). Clause 8.3.3.5 describes the following common aspects of these two M2M Service Bootstrap procedures:

- Description of the Bootstrapping Credentials (clause 8.3.3.5.2).
- Description of the architecture applicable to these bootstrapping processes (clause 8.3.3.5.3), including a description of the architecture entities and architecture interfaces.
- Description of the MSBF Certificate Status Verification methods (clause 8.3.3.5.4).

8.3.3.5.2 Bootstrapping credentials when Using Device Certificates

Bootstrapping credentials shall be pre-provisioned to M2M Device/Gateway. The bootstrapping credentials may be provisioned at manufacturing time. These bootstrapping credentials enable mutual authentication between the MSBF and the M2M Device/Gateway at bootstrapping time.

The bootstrapping credentials shall include:

- At least one globally unique Pre-provisioned-ID (assigned by M2M Device/Gateway manufacturer).

  A unique private key for digital signatures, and the corresponding device certificate containing public key. This certificate shall conform to the details provided in [1].

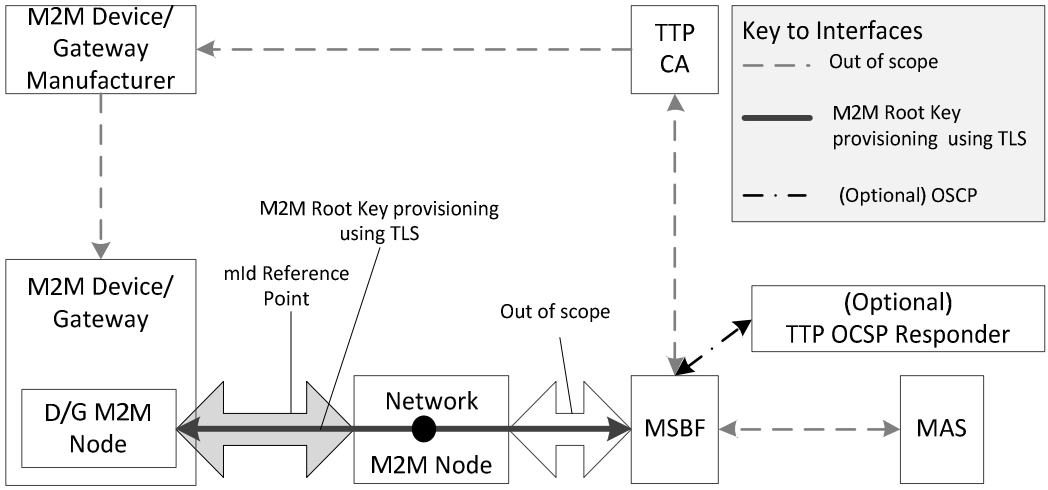
  One or more Trusted Third Party (TTP) certificate(s) to be used as trusted root certificate(s). This certificate(s) allows verification of MSBF certificates. These certificates shall conform to the details provided in [1].

- The M2M Device/Gateway may need provisioning with additional data, depending on the MSBF Certificate Status Verification method to be applied by the D/G M2M Node (see clause 8.3.3.5.4).

8.3.3.5.3 Architecture

8.3.3.5.3.1 Architecture Overview

Figure 8.10 shows the M2M Service Bootstrap architecture when using TLS.



NOTE: The line through the Network M2M Node indicates that the Network M2M Node passes messages between the D/G M2M Node and MSBF with some procedure-dependent processing at the Network M2M Node.

Figure 8.10: Architecture for TLS/Certificates-Based M2M Service Bootstrap Procedures

8.3.3.5.3.2 Architecture Components

The roles of the architecture components in TLS/Certificates-Based M2M Service Bootstrap procedures are as follows.

- The roles of the MAS are as for other bootstrapping methods.
- The D/G M2M Node performs the same roles as for other bootstrapping methods.
- The role of the M2M Device/Gateway Manufacturer (with respect to these procedures) is pre-provisioning bootstrapping credentials into the M2M Device/Gateway containing the D/G M2M Node as described in clause 8.3.3.5.2.
- The role of the Network M2M Node depends on the particular bootstrapping procedure being used. In both cases, the Network M2M Node forwards messages between the D/G M2M Node and MSBF, and the Network M2M Node does not perform any processing on the TLS layer messages.
- The TTP Certificate Authority (TTP CA) issues certificates to the M2M Device/Gateway Manufacturer and MSBF as described in clause 8.3.3.5.2.
- The (optional) TTP Online Certificate Status protocol (OCSP) Responder provides OCSP services to the D/G M2M Node as described in Step 1 in clause 8.3.3.5.4.2.

8.3.3.5.3.3 Architecture Interfaces

The interface between the M2M Device/Gateway manufacturer and the M2M Device/Gateway containing the D/G M2M Node is out of scope. This interface shall support initial provisioning of bootstrapping credentials and optionally for remote update of bootstrapping credentials.

The interface between the TTP CA and M2M Device/Gateway manufacturer is not specified in the present document. This interface shall support providing relevant portions of the bootstrapping credentials from the TTP to M2M Device/Gateway manufacturer.

The interface between the MAS and MSBF is out-of-scope. In many cases, the MSBF may be internal to the M2M SP, but this is not a requirement as the MSBF may be a third party trusted by the M2M SP to provide bootstrapping. This interface shall support providing D/G M2M Node information from the MAS to the MSBF, and for the MSBF to send to the MAS the M2M-Node-ID and Kmr.

The communication between the D/G M2M Node and Network M2M Node shall use reference point mId. This reference point is part of the path for messages passed between D/G M2M Node and MSBF.

The reference point between the Network M2M Node and MSBF is not specified in the present document. This reference point is part of the path for messages passed between D/G M2M Node and MSBF.

The interface between TTP CA and MSBF shall support providing the MSBF with MSBF certificates. This interface is not specified in the present document.

The interface between the MSBF and (optional) TTP OCSP Responder performs two roles. This interface may allow the MSBF to obtain verification (from the TTP OCSP Responder) of the status of the M2M Device/Gateway's certificate. This interface shall also allow the D/G M2M Node to obtain verification of the status of the MSBF certificate - in this case the MSBF submits the request to the TTP OCSP Responder on behalf of the D/G M2M Node, and the MSBF forwards the response (from the TTP OCSP Responder) back to the D/G M2M Node. This interface shall use OCSP [33].

8.3.3.5.3.4 Trusted third party

The trusted third party (TTP) shall issue public key certificates either to M2M Devices/Gateways directly, or via M2M Device/Gateway manufacturers. In the latter case, the TTP shall issue Certificate Authority (CA)-level certificates to M2M Device/Gateway manufacturers who in turn issue certificates to M2M Devices/Gateways.

The TTP shall provide resources for M2M service providers to check the current status of device certificates (for example, using OCSP or certificate revocation lists). Two methods are provided in clause 8.3.3.5.4:

- D/G M2M Nodes may access the TTP OCSP Responder (if provided by the TTP) through the MSBF, (see clause 8.3.3.5.4.2); or
- D/G M2M Nodes may use a Certificate Revocation List residing on the D/G (see clause 8.3.3.5.4.3).

MSBF certificates may be issued by M2M SP themselves who have a trust relationship with the TTP, or by some other entity trusted by the TTP.

8.3.3.5.4 MSBF Certificate Status Verification Methods

8.3.3.5.4.1 Overview

The present document supports two methods for the D/G M2M Node to verify of the status of the MSBF certificate:

- Online Certificate Status Protocol (OCSP) Method, clause 8.3.3.5.4.2.
- Certificate Revocation List (CRL) Method, clause 8.3.3.5.4.3.

8.3.3.5.4.2 Online Certificate Status Protocol (OCSP) Method

Requirement Status:

- The support for this method is optional to implement on the D/G M2M Node.
- The support for this method is mandatory to implement on MSBF if the MSBF supports D/G M2M Node that will apply this method.

Preconditions:

- This method can only be supported by the D/G M2M Node if:
  - the TTP provides an TTP OCSP Responder; and
  - a certificate for the TTP OCSP responder is provisioned on the M2M Device/Gateway. The set of acceptable profiles for these certificates is provided in [1].

Procedures:

To apply the OCSP Method, the OCSP shall be integrated into the TLS handshake as specified in [36] for TLS v1.1 or [44] for TLS v1.2. The following informative text describes the steps from [36], [44]. Where there are discrepancies between this text and [36], [44], the later texts take precedence.

- The D/G M2M Node includes the OCSP parameters in the TLS extension `CertificateStatusRequest` field of the TLS ClientHello message.
- If the MSBF receives the TLS extension `CertificateStatusRequest` field then the MSBF extracts the OCSP parameters.
- The MSBF forms an OCSP Request using its certificate and the parameters received from the D/G M2M Node, and the MSBF sends the OCSP Request to the indicated OCSP Responder.
- The OCSP Responder processes the OCSP Request, determines the status of the MSBF Certificate and replies with an OCSP Response.
- The MSBF encapsulates the OCSP Response in a TLS CertificateStatus message (this message is defined in TLS Extensions [36], [44]). This TLS CertificateStatus message is sent back to the D/G M2M Node along with other TLS handshake messages.
- The D/G M2M Node extracts the OCSP Response from the TLS CertificateStatus message, and processes the OCSP Response using the provisioned TTP OCSP Responder certificate (as per [33]) to obtain a verified status of the MSBF certificate.

8.3.3.5.4.3 Certificate Revocation List (CRL) Method

Support Requirements: This method is optional to implement on the D/G M2M Node. This method does not impact on any other entities in the architecture.

Preconditions: A valid CRL shall be present on the M2M Device/Gateway containing the D/G M2M Node at the time of verifying the MSBF certificate status. The process for updating the CRL on the M2M Device/Gateway is outside the scope of the present document.

Procedure: To apply the CRL Method, the D/G M2M Node shall apply the processes defined in [49] to the MSBF certificate, using the most recent CRL present on the M2M Device/Gateway.

8.4 M2M Service Connection procedures

This clause describes the M2M Service Connection procedures between a D/G M2M Node and the Network Domain of the M2M service provider for which an M2M Root Key (Kmr) has been provisioned in the D/G M2M Node.

8.4.1 Overview

The M2M Service Connection procedure shall be performed according to the following steps:

- Mutual mId end point authentication.
- M2M Connection Key agreement.

- Reporting of Integrity Validation security attributes for those M2M Service Providers that support IVal (see annex C):
  - The MAS provides the NSCL with additional security attributes regarding the ability of the D/G to perform and report IVal (this information enables the NSCL to perform policy-based access control), as well as the public key corresponding to the private key that resides in the Secured Environment on the device that is used to sign the IVal results.
- Optional establishment of secure session using encrypted communication over mId.

After successful establishment of M2M Service Connection, D/GSCL registration with NSCL according to clause 9.3.2.6.2 and subsequent RESTful procedures between D/GSCL and NSCL may be performed.

This M2M Service Connection procedure requires that M2M Service Bootstrap has already been successfully completed for the M2M SP that the targeted Network M2M Node is associated with.

Key agreement policies are out of scope of the present document.

Secure communication over mId is achieved via encryption and integrity protection, using Kmc. More specifically, upon mutual authentication and key agreement, Kmc may be used to protect the subsequent SCL registration message exchanges over mId.

8.4.2 M2M Service Connection procedure based on EAP / PANA

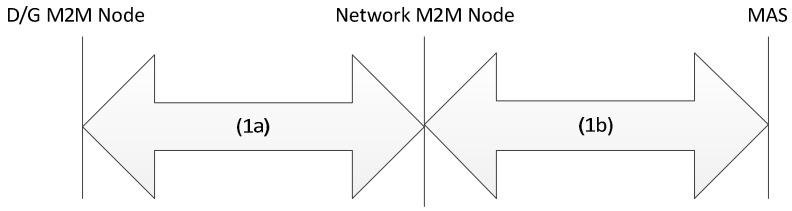


Figure 8.11: M2M Service Connection procedure

This procedure shall be performed according to the following steps:

Step 1:

- The D/G M2M Node and the MAS shall authenticate each other via the Network M2M Node.
  - Before entering this phase, the D/G M2M Node is assumed to be provisioned with an M2M-Node-ID and the Kmr. These credentials shall be used by the MAS and D/G M2M Node for mutual authentication.
- The authentication procedure may be carried out by using EAP between the D/G M2M Node and the MAS via the Network M2M Node.
  - Step 1a: EAP is carried over PANA between the D/G M2M Node and the Network M2M Node.
  - Step 1b: EAP is carried over AAA protocol (e.g. RADIUS, Diameter) between the Network M2M Node and the MAS An EAP method that can use the aforementioned credentials is used over EAP.
- Attributes specifically needed for the M2M Service Connection procedure are defined as PANA AVPs and AAA attributes and carried as such.
- At the end of successful EAP-based mutual authentication and delivery of MSK to the NSCL, Kmc shall be generated by the NSCL and the D/G SCL according to the following formula:

$$Kmc = Hash(MSK, \text{"ETSI M2M Connection Key"} \mid D/GSCLD/GSCL\text{-}ID)$$

Hash is a one-way keyed hash function specified in TS 102 921 [1].

MSK is the EAP Master Session Key that is available on the D/G M2M Node and the Network M2M Node at the end of successful EAP authentication.

D/GSCLD/GSCL-ID is the service capability layer ID of the D/G SCL.

8.4.3 M2M Service Connection procedure based on TLS-PSK

8.4.3.1 Overview

This clause describes a mechanism of using TLS-PSK [50], [41], [43], [47], for establishing Kmc between a D/G M2M Node and NSCL with the assistance of a MAS with whom the D/G M2M Node has already established Kmr. This M2M Connection Procedure may only be used when mId is secured using object security or channel security (see clause 8.5).

This procedure makes the following assumptions:

- The MAS shall share a valid Kmr associated with the D/G M2M Node that wishes to be authenticated. If the D/G M2M Node and MAS do not share a valid Kmr associated with this D/G M2M Node, then an M2M Bootstrapping Procedure shall be performed first to establish a valid Kmr.
- The D/G M2M Node may wish to establish M2M Service Connections to multiple Network M2M Nodes.
- The D/G M2M Node shall know the URI of the Network M2M Node(s) to which it wishes to authenticate.
- To support this procedure, the Network M2M Node shall have a dedicated IP address for this procedure (that is, M2M Service Connection procedure based on TLS-PSK). This dedicated IP address enables the Network M2M Node to identify packets intended to be used for this procedure, so the Network M2M Node can act accordingly.
    - The Network M2M Node shall perform network address translation to IP packet containing TLS layer messages received from the D/G M2M Node and MAS, to direct those IP packets to the correct destination. The Network M2M Node shall not perform any processing on the TLS layer messages. That is, the Network M2M Node shall act as a NAT [34].
    - The M2M SP shall ensure that the D/G M2M Node is directed to the correct IP address for this procedure. Further details are found in [1].
- The interface between the Network M2M Node and MAS is not specified in the present document: for example, an AAA protocol can be used for this interface.
- The communication between Network M2M Node and MAS shall be secured.
- The MAS shall know the URI(s) associated with the Network M2M Node.

A high level call flow for this procedure is shown in clause 8.4.3.2.

The assumptions listed above result in communication between D/G and MAS for the purpose of initial authentication and key agreement.

8.4.3.2 High Level Call Flow

The following call flow describes the procedure. Details internal to TLS are not shown. The call flow is shown in Figure 8.12.

1) The D/G M2M Node and MAS shall perform TLS-PSK handshake [50], [47], communicating via an Network M2M Node. The D/G M2M Node shall send its TLS layer messages to the IP address at the Network M2M Node that the Network M2M Node has dedicated for "M2M Service Connection procedure based on TLS-PSK". The Network M2M Node shall not process the contents of TCP packets using this IP address (the contents are TLS layer messages), but shall simply performs network address translation to direct these packets to and from the MAS. That is, the Network M2M Node shall act as a NAT [34].

a) The present document places no requirements on the psk_identity_hint [50], [47], in the ServerKeyExchange message (sent by the MAS).

b) The D/G M2M Node shall set the psk_identity [50], [47](in ClientKeyExchange message [41], [43]) to be the M2M-Node-ID (which is known to the MAS).

c) The D/G M2M Node and MAS shall set PSK [50] to the value of Kmr shared by the D/G M2M Node and MAS.

d) The ciphersuite for this use of TLS-PSK shall comply with [1].

2) TLS shall secure the communication between the D/G M2M Node and MAS. The D/G M2M Node and MAS repeat steps 2a and 2b for each Network M2M Node that the D/G M2M Node wishes to be authenticated to:

a) The D/G M2M Node shall provide the URI of the Network M2M Node that the D/G M2M Node wishes to be authenticated to. The formatting of this message shall comply with [1].

b) The MAS shall perform the following:

i) The MAS generates an M2M-Connection-ID as specified in [1]

ii) The MAS shall assign the M2M Connenection -lifetime.

iii) The shall MAS generate fresh, random secret key Kmc.

The MAS shall send to the D/G M2M Node the M2M-Connection-ID, M2M-Connection-lifetime and Kmc and any other appropriate attributes. The formatting of this message shall comply with [1].

The D/G M2M Node and MAS shall close the TLS session. The MAS shall store the D/G M2M Node-ID, Network M2M Node-URI, M2M Connection-lifetime, Kmc, and any other appropriate attributes, indexed by M2M-Connection-ID.

3) To be authenticated to one of these NSCL (and establish secure communication with the NSCL), the D/GSCL provides the M2M-Connection-ID to the NSCL as part of the object-security or channel security procedure (see clause 8.5).

NOTE: The M2M Service Connection Procedure based on TLS-PSK does not support mId security relying on access network security.

4) If the Network M2M Node has already communicated with the MAS regarding this M2M-Connection-ID, then the Network M2M Node shall skip to Step 7. Otherwise, the Network M2M Node shall parse the M2M-Connection-ID to determine the MAS-FQDN. The Network M2M Node then forwards a request to the MAS containing the M2M-Connection-ID.

5) The MAS verifies that the Network M2M Node is associated with the Network M2M Node-URI associated with the M2M-Connection-ID. If so, the MAS shall reply to the Network M2M Node with M2M-Node-ID, M2M-Connection-lifetime (Optional), Kmc and any other appropriate attributes.

6) The Network M2M Node shall respond to the D/G M2M Node and the D/G M2M Node and Network M2M Node then use key Kmc for securing mId, as specified in clause 8.5. The key hierarchy for generating keys from Kmc is described in clause 8.2.2.1.

7) Steps 3-6 may be repeated as often as required.

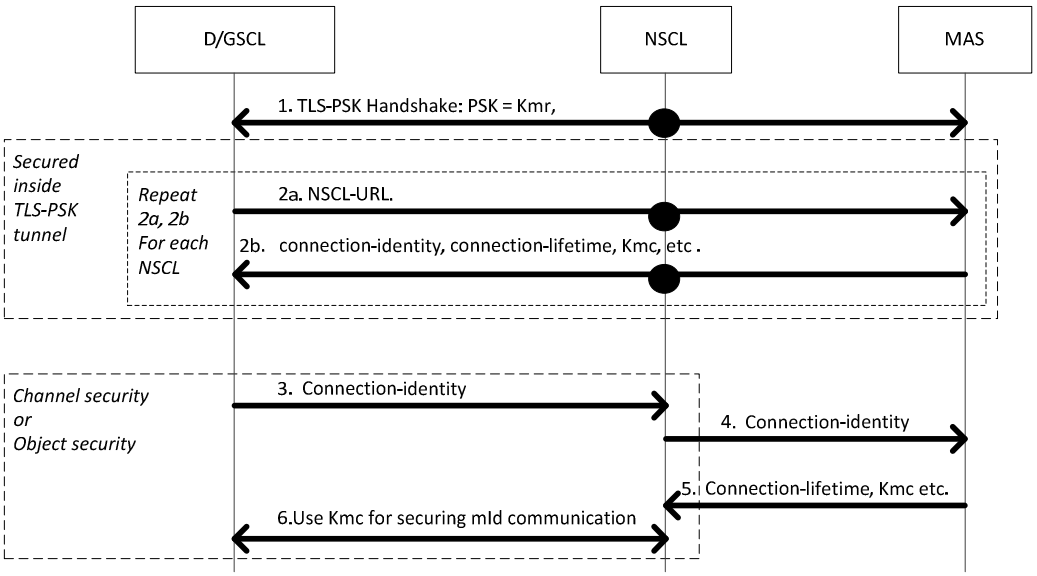


Figure 8.12: High Level Call flow for M2M Service Connection procedure based on TLS-PSK

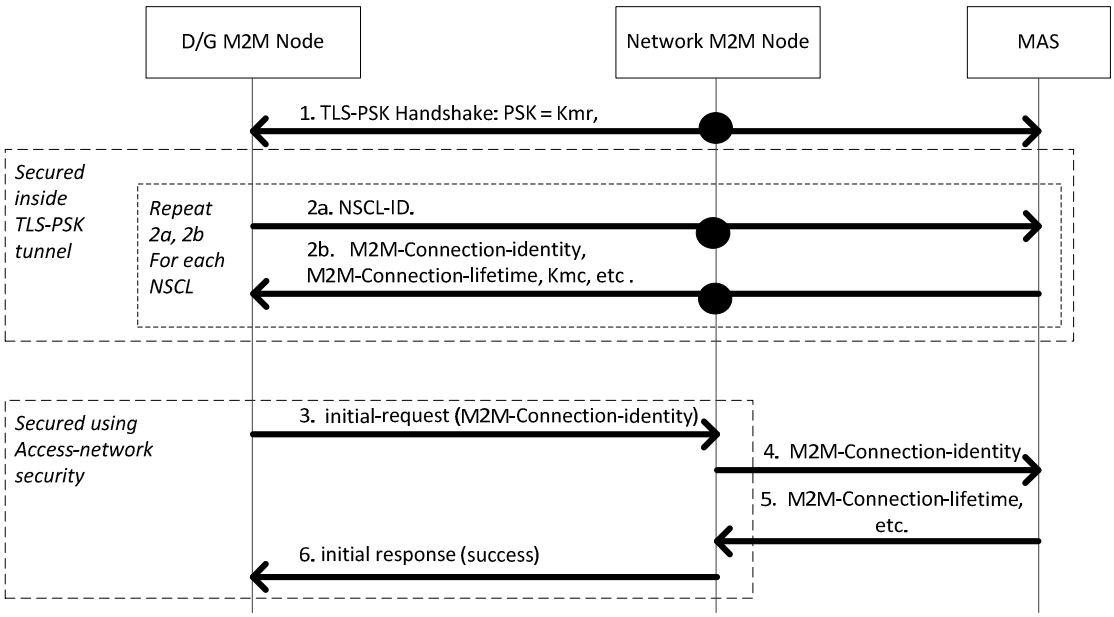


Figure 8.13: High Level Call flow for M2M Service Connection procedure based on TLS-PSK when using access network security for mId Data Security

8.4.4 M2M Service Connection procedure based on GBA

In scenarios where the Access Network Provider and the M2M Service Provider have a trust relationship (including the case that they are actually the same entity), the long term key stored in USIM/ISIM/CSIM/(R)- UIM can be used in GBA procedures as specified by TS 133 220 [29], 3GPP2 S.S0109 [51] for performing mutual authentication and key agreement.

Figure 8.14 depicts a M2M Service Connection procedure where GBA is run for mutual authentication and key agreement.

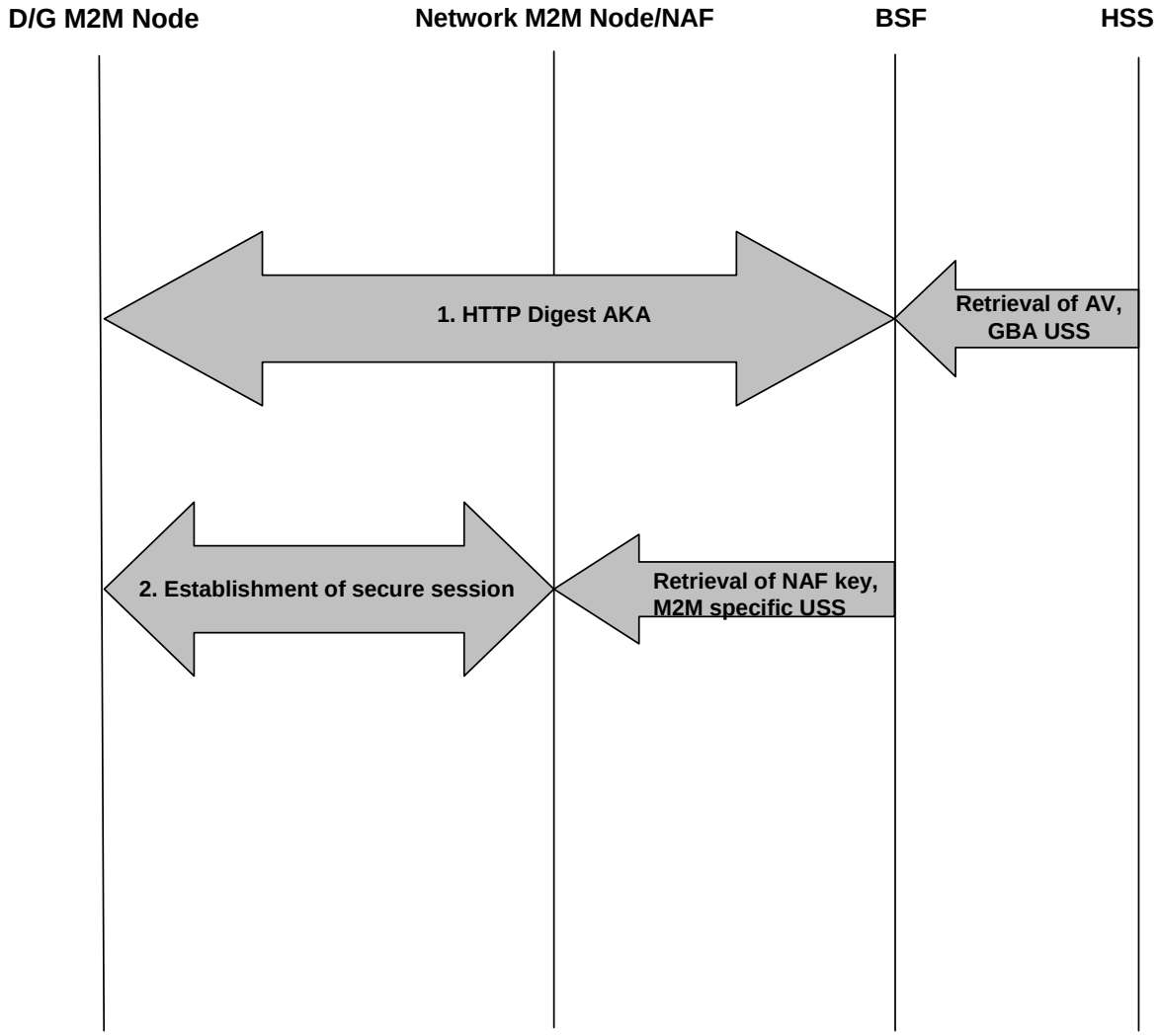


Figure 8.14: M2M Service Connection procedure based on GBA

The high-level procedure shall be comprised of the following steps:

Step 1: The D/G M2M Node shall carry out HTTP Digest AKA over the GBA Ub reference point towards the BSF, using an authentication vector (AV) the BSF fetched from the HSS. The BSF also retrieved the GBA User Security Settings (USS) from the HSS which may contain M2M specific security settings The details of this step shall comply with [29] and [51]. Annex A of [5] gives signalling flows.

Step 2: The D/G M2M Node and the Network M2M Node/NAF shall use the NAF-specific key to establish a secure session between them, by running TLS-PSK. The details of this step shall comply with [5].

In this step the D/G M2M Node sends the GBA bootstrapping transaction identifier (B-TID) obtained from step 1 to the Network M2M Node /NAF. The Network M2M Node/NAF uses the B-TID to retrieve the NAF-specific key and optionally the M2M specific USS from the BSF, over the Zn reference point as specified in [51]. As a result, the D/G M2M Node and the Network M2M Node/NAF share the NAF-specific key which shall be used as the M2M Connection Key.

8.5 mId Security

The mId Reference Point shall support data origin authentication, integrity and replay protection, confidentiality, and privacy. There are three distinct ways the mId may be secured:

1) Access network layer security

   In case the underlying access network is already cryptographically or physically secured, and there is a pre-established trust between the access network and the M2M network, then it is possible to use this underlying security as a substitute for the M2M layer security. Alignment between the end-points at the access network layer and the M2M network layer is necessary for enabling such a substitution. A careful study shall be conducted before choosing this option.

2) Channel security

   A secure communication channel may be built between the D/G M2M Node and the Network M2M Node, and used to protect all information exchanged over mId. Supported protocols are specified in TS 102 921 [1] Kmc shall be used as a shared secret key between the two end-points for performing end-point authentication. Once a secure channel is established, mId protocols shall be carried over that channel for achieving security.

   Secure channel can be built only after the M2M Service Connection procedure takes place.

3) Object security

   An M2M implementation may rely on securing data at the object (i.e. protocol payload) level.

   When channel security is used, each piece of data transferred over the mId is subjected to the same level of security. And that level is determined by the highest need among all data. For example, if just a small part of data requires encryption, then the whole data communication is encrypted at all times. The way to reduce this potential inefficiency is to use security at the same layer as data is transmitted, hence gaining finer-granularity security over the data. Each data element can be individually integrity protected and encrypted without any regard to how other data is treated.

At least a channel or object security solution shall be implemented on the D/G M2M Node and the Network M2M Node. At least one of the three options provided shall be used for protecting mId. More than one approach may be combined in a given deployment. For example, object security can be used with selected objects for achieving confidentiality and privacy over a channel that provides integrity protection.

9 M2M Resource Management and Procedures

9.1 Introduction

9.1.1 Usage of resources in a RESTful architecture

The procedures described below adopt a RESTful architecture style. This style governs how M2M Applications (DA, GA, NA) and/or M2M SCL are exchanging information with each other. The properties of a RESTful style are not described here, please refer to [i.1]. A RESTful architecture is about the transfer of representations of uniquely addressable resources. ETSI M2M standardized the resource structure that resides on a SCL.

In a very simplistic view, imagine that certain resources are buckets that can hold some application specific data. These buckets - as far as the scope of an M2M service layer is concerned - reside in the respective SCL. The buckets have certain properties and are structured as suggested in more detail in the subsequent clauses.

To illustrate a very basic use of this mediator function of the M2M SCL, a simple example shall be described here. An application (DA) on an M2M Device that is not always connected wants to send some data to another application (NA) on the network by means of the M2M SCL. DA would write data to a resource in the NSCL and NA would read that resource. If configured accordingly, the NA could also be notified by the NSCL upon the writing (update) of the resource by the DA, in order to facilitate synchronization between DA and NA. Figure 9.1 is meant to illustrate that process of the data flow and not the operation flow.

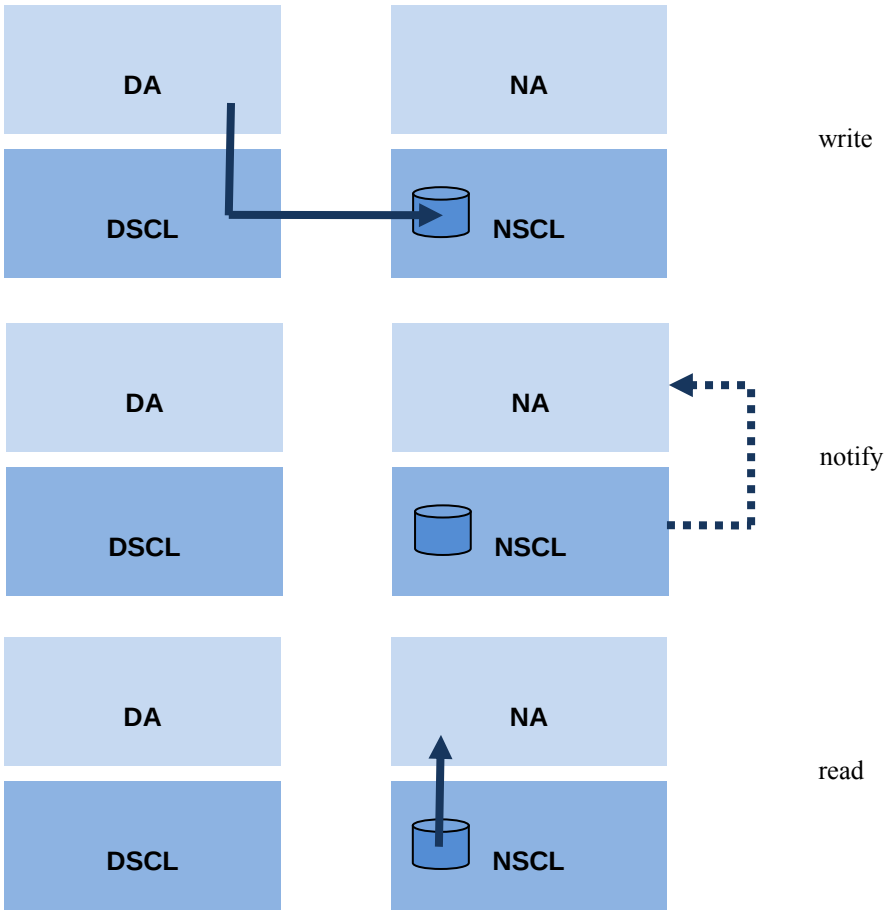


Figure 9.1: Simple example for use of SCL resources to exchange data

When handling resources in a RESTful architecture, there are four basic methods - so called "verbs" - that could be applied to resources:

- CREATE: Create child resources.
- RETRIEVE: Read the content of the resource.
- UPDATE: Write the content of the resource.
- DELETE: Delete the resource.

These methods are referred to as the CRUD methods below. In addition to these basic methods in a RESTful architecture, it is often also useful to define verbs actions that might not directly map to one of the specific method. Moreover a definition of these particular verbs helps the readability of the present document if the verb is chosen appropriately. The additional verbs introduced are:

- NOTIFY: used to indicate the operation for reporting a notification about a change of a resource as a consequence of a subscription. This verb would either map to a response of a RETRIEVE method in case that the long polling mechanism is used, or to an UPDATE method in case that the asynchronous mechanism is used.
- EXECUTE: for executing a management command/task which is represented by a resource. This verb corresponds to an UPDATE method without any payload data.

9.1.2 Definitions

This clause provides definitions that are used for describing the resource procedures detailed in clauses 9.2 and 9.3.

Resource: is a uniquely addressable entity in the RESTful architecture. A resource has a representation that shall be transferred and manipulated with the verbs. A resource shall be addressed using a Universal Resource Identifier (URI).

Sub-Resource: also called child resource. It is a resource that has a containment relationship with the addressed (parent) resource. The parent resource representation contains references to the children. The lifetime of the sub-resource is linked to the parent's resource lifetime.

Attribute: is meta-data that provides properties associated with a resource representation.

Attribute-Type: attributes are distinguished by the following types:

RW	read/write by client
RO	Read-Only by client, set by the server
WO	Write-once, can be provided at creation, but cannot be changed anymore

Note that if an attribute is RW, it does not mean that the Issuer can set it to any value. Some values may be restricted by the Hosting SCL, e.g. due to policies. For example, an expiration time may be written by the Issuer, but it is treated as a suggestion by the Hosting SCL. The Hosting SCL is free to change (i.e. lower) the expiration time. If the Hosting SCL changes anything it shall send back a full representation of the resource as it is created, i.e. a success response with a body.

Issuer: is the actor performing a request. An issuer shall either be an Application or a SCL.

Announced Resource: the content of this resource refers to a resource hosted by the Hosting SCL (Master/original Resource). The purpose of this resource is to facilitate a discovery of the original resource, so that the issuer of the discovery does not have to contact all SCLs in order to find the resource. For detailed description about this resource, refer to clause 9.2.1.14.

Local SCL: The Local SCL is the SCL where an Application or a SCL shall register to. It is the first SCL that receives the request from the original issuer of the request (either an Application or a SCL):

- if the NA is the original issuer, the Local SCL is the NSCL;
- if the GA is the original issuer, the Local SCL is the GSCL;
- if the DA in a D device is the original issuer, the Local SCL is the DSCL;
- if the DA in a D' device is the original issuer, the Local SCL is the GSCL;
- if the DSCL in a D Device is the original issuer then the local SCL is the NSCL or the GSCL.

Hosting SCL: The SCL where the addressed (Master/original Resource) resource resides.

NOTE: In some cases the hosting SCL also act as Local SCL, this is valid in case of a registration.

Announced-to SCL: a SCL that contains the announced resource (a resource could be announced to multiple SCLs).

Receiver: it represents the actor that receives a request from an issuer. A receiver shall be a SCL or an Application.

9.2 Resource structure

9.2.1 Types of resources to be used in a SCL

This clause introduces the main types of resource used in a SCL. Since all of these resources will have to be addressed in some way and since there are relationships between them (like a parent-child containment relationship), a hierarchical tree structure for modelling their structure and relationships is presented next is clause 9.2.3.

9.2.1.1 SclBase Resource

The sclBase resource shall contain all other resources of the hosting SCL. An sclBase resource is the root of all other resources it contains. The sclBase resource shall represented by an absolute URI. All other resources hosted in the SCL shall also be identified by a URI.

For example, a specific sclBase resource identifying an Network SCL could be:
"`http://m2m.myoperator.org/some/arbtrary/base/`".

An example of a URI identifier of a container resource hosted by this network SCL could be:
"`http://m2m.myoperator.org/some/arbtrary/base/containers/myExampleContainer`".

9.2.1.2 SCL Resource

SCL resource shall represent an associated (remote) SCL that is authorized to interact with the hosting SCL. In order to be authorized to interact with the hosting SCL, the remote SCL has to go through a M2M service registration procedure. An SCL resource is created as a result of a successful registration of the remote SCL with its local SCL or vice-versa. SCL resource shall store context information about the registered SCLs. Registered SCL resource shall contain other resources as described in the tree structures in clause 9.2.3.4.

9.2.1.3 Application Resource

Application resource shall store information about the Application. Application resource is created as a result of successful registration of an Application with the local SCL. Applications shall only register to their local SCL.

9.2.1.4 AccessRight Resource

AccessRight resource shall store a representation of permissions. An accessRight resource is associated with resources that shall be accessible to entities external to the hosting SCL. Basically, they control "who" (permissionHolder) is allowed to do "what" (permissionFlag). Moreover, they can be used in the privacy protection.

9.2.1.5 Container Resource

Container resource is a generic resource that shall be used to exchange data between applications and/or SCLs by using the container as a mediator that takes care of buffering the data. Exchange of data between applications (e.g. on device and network side) is abstracted from the need to set up direct connections and allows for scenarios where both parties in the exchange are not online at the same time.

9.2.1.6 LocationContainer Resource

A LocationContainer resource shall represent a container for the location information of a M2M entity (e.g. M2M Device/Gateway). The location information may be generated by a Device/Gateway application or provided by location server in the network domain.

9.2.1.7 Group Resource

Group resource shall be used to define and access groups of other resources.

For example, a group resource could be used to write the same content to a group of M2M container resources (this allows a DA to write the same data to many container resources in a NSCL. The data is only sent once to the NSCL, while letting the NSCL replicate it to different container resources. This is a more optimal use of the dIa/mId reference point).

When a group resource is a member of a second group resource, it can be referred as a sub-group of the second group resource.

9.2.1.8 Subscription Resource

Subscription resource shall be used to keep track of status of active subscription to its parent resource. A subscription represents a request from the Issuer to be notified about modifications on the parent resource.

9.2.1.9 M2MPoC Resource

The M2MPoC resource shall represent information maintained in the NSCL on how to reach a DSCL or GSCL via a specific access network. The device or gateway maintains this information in the NSCL by creating, updating, or deleting this resource when their point of attachment changes.

9.2.1.10 MgmtObj Resource

A MgmtObj resource holds the management data which represents a certain type of M2M remote entity management function. For a remote entity (i.e. M2M Device/Gateway) multiple mgmtObj resources may be created on NREM for different management purposes.

9.2.1.11 MgmtCmd Resource

A MgmtCmd resource shall be only used to model non-RESTful management commands, i.e. BBF TR-069 Remote Procedure Call (RPC) methods, as listed in Table 9.1. This resource represents the RPC methods in a RESTful manner. With such RESTful modelling, a MgmtCmd (i.e. a RPC) shall be triggered by an NA using the RESTful verb UPDATE. If supported by the specific BBF TR-069 procedure, a triggered MgmtCmd may be cancelled before it finishes by an NA using UPDATE verb or a DELETE verb.

Table 9.1: BBF TR-069 RPC to be modelled as MgmtCmd Resource

BBF TR-069 RPC Methods	Meaning
FactoryReset	Used by the ACS [10] to reset the CPE [10] to its factory default state.
Reboot	Used by the ACS to cause the CPE to reboot.
Upload	Used by the ACS to cause the CPE to upload a specified file to the designated location. It can be cancelled by BBF TR-069 CancelTransfer RPC.
Download	Used by the ACS to cause the CPE to download a specified file from the designated location. It can be cancelled by BBF TR-069 CancelTransfer RPC.
ScheduleDownload	Used by the ACS to cause the CPE to download a specified file from the designated location and apply it within either one or two specified time Windows. It can be cancelled by BBF TR-069 CancelTransfer RPC.
ScheduleInform	Used by the ACS to request the CPE to schedule a one-time Inform method call sometime in the future.
ChangeDUState	Used by an ACS to trigger the explicit state transitions of Install, Update, and Uninstall for a Deployment Unit (DU), i.e. installing a new DU, updating an existing DU, or uninstalling an existing DU.

9.2.1.12 AttachedDevices Resource

An AttachedDevices resource shall be used to collect the management information of all M2M D' devices that are attached to a M2M Gateway. It shall reside under a G <*scl*> resource created in the remote NSCL.

9.2.1.13 AttachedDevice Resource

An AttachedDevice resource shall be used to represent each M2M D' device that is attached to a M2M Gateway. The resource lives only in the NSCL and it shall reside under the AttachedDevices resource of the corresponding M2M Gateway.

9.2.1.14 Announced Resource

An announced resource shall point to the original resource hosted in another SCL. The announced resource is an actual resource which consists of only a limited set of attributes, which are the searchStrings, the link to the original resource and the access right. The purpose of the announced resource is to facilitate a discovery of the original resource when querying the announced-to SCL, so that the issuer of the discovery does not have to contact all SCLs in order to find the resources. An announced resource itself shall only be visible when it is directly accessed via its full URI. During discovery a direct reference to the original resource shall be returned. Only locally created resources can be announced.

Removing an announced resource, for example due to deregistration of an SCL (which correspond to a removal of the parent SC L resource), does not remove the original resource, but does remove all the children of announced resource Reversely, when the original resource is removed, it is the responsibility of the original SCL, where the original resource is hosted, to remove the announced resource. If this is not done (e.g. because the original SCL is offline), the announcement resource shall be removed when it expires.

NOTE: There are collections with:
1) only real resources;
2) a mix of real resources and announced resources;
3) only announced resources.

A resource of the same type (e.g. <*application*> resources) might have different content depending on where in the tree it is located. There are different ways of representing this, i.e. either by defining different resources or by defining one collection resource that contain both types of children. The latter solution is chosen, but it is indicated in each case whether which child resources are allowed.

9.2.1.15 NotificationChannel Resource

A NotificationChannel resource shall be used by non-server capable client to receive asynchronous notifications. The notification channel is prepared to handle several mechanisms on how to receive these asynchronous notifications. However, currently only one mechanism is fully specified, which is the so-called "long polling" mechanism. This method is based on the server not responding to requests until a notification needs to be sent (or until a timeout occurs).

9.2.1.16 Discovery Resource

A discovery resource shall be used to allow discovery. It is used to retrieve the list of URI of resources matching a discovery filter criteria. It does not represent a real resource in the sense that it does not have a representation and it shall never be cached.

9.2.1.17 Collection Resource

This resource represents an abstract concept that is applicable to various resources in the resource structure. For details on the collection resources, see clause 9.3.

A collection resource normally has its own associated accessRightID and allows subscriptions on modification in the collection resource.

I.e. when resources contain a collection of similar sub-resources, this is modelled as a collection resource. There are several collection resources identified, e.g. the SCL resource mentioned above contains collection resources for <*group*> resources, for <*container*> and <locationContainer> resources, for <*application*> resources, for <*accessRight*> resources and for <mgmtObj> and <mgmtCmd> resources. A collection can container local resources and/or the corresponding announced resources. A collection resource representation contains the sub-resources by reference and it may also contain attributes.

9.2.2 Common attributes

Many of the attributes of the resources described in the present document are common. Those attributes are described here once in order to avoid duplicating the description for every resource that contains it.

Attributes that are only used in one or two resource types are described only in the section for that resource.

Table 9.2

Name	Description
accessRightID	URI of an access rights resource. The permissions defined in the accessRight resource that is referenced determine who is allowed to access the resource containing this attribute for a specific purpose (retrieve, update, delete, etc.). If a resource type does not have an accessRightID attribute definition, then the accessRights for resources of that type are governed in a different way, for example, the accessRight associated with the parent may apply to a child resource that does not have an accessRightID attribute definition, or the permissions for access are fixed. Refer to the corresponding procedures to see how permissions are handled in these cases. If a resource type does have an accessRightID attribute definition, but the (optional) accessRightID attribute is not set, or it is set to a value that does not correspond to an valid, existing, accessRight resource, or it refers to an accessRight resource that is not reachable (e.g. because it is located on a remote SCL that is offline or not reachable), then the system default access permissions shall apply. The system default access permissions grant all permissions (i.e. the full set of permissionsFlags) to the following permission holders depending on the prefix of URI of the resource. The permissionHolders for prefixes from most specific to least specific are as follows: <*sclBase*>/*scls*/<*scl*>/*applications*/<*applicationAnnc*>: the hosting SCL, the SCL corresponding to the <*scl*> resource and the Application corresponding to the <*applicationAnnc*> resource shall be the permissionHolders. <*sclBase*>/*scls*/<*scl*>: the hosting SCL and the SCL corresponding to the <*scl*> resource shall be the permissionHolders. <*sclBase*>/*applications*/<*application*>: the hosting SCL and the Application corresponding to the <*application*> resource shall be the permissionHolders. <*sclBase*>: the hostingSCL shall be the permissionHolder.
announceTo	In a request on mIa or dIa, this is interpreted as the list of the SCLs that the SCL will try to announce to on behalf of the requestor. In responses, the list indicates the actual list of resources to which the resource is announced at the moment. If this attribute is not provided in requests on the mIa or dIa, the local SCL will decide where the resource will be announced.
creationTime	Time of creation of the resource.
expirationTime	Absolute time after which the resource will be deleted by the hosting SCL. This attribute can be provided by the issuer, and in such a case it will be regarded as a hint to the hosting SCL on the lifetime of the resource. The hosting SCL can however decide on the real expirationTime. If the hosting SCL decides to change the expirationTime attribute value, this is communicated back to the issuer. The lifetime of the resource can be extended by providing a new value for this attribute in an UPDATE verb. Or by deleting the attribute value, e.g. by not providing the attribute when doing a full UPDATE, in which case the hosting SCL can decide on a new value.
filterCriteria	This are criteria that filter the results. They can either be used in a GET (as query parameters) or in a subscribe.
lastModifiedTime	Last modification time of a resource.
link	URI of the related remote resource. In an announced resources, this is the URI of the announcing resource. In an <*scl*> resource, this is the URI of the <*sclBase*> resource of the registered SCL.
searchStrings	Tokens used as keys for discovering resources.

9.2.3 Tree structure modelling relationship of different resource types

9.2.3.1 Overview

The present document uses a tree representation for describing how the different types of resources relate to each other. This is essential for deriving a meaningful way to navigate to the different resources and understand their use. The tree structure described in this clause does not mandate a physical implementation for data storage however the access to resources shall be done by traversing the tree and using resource links as per the tree structure described in the remaining of clause 9.2.3 and related subclauses. The structure described in this clause applies to all SCLs, i.e. the same structure applies to resources in the NSCL, the GSCL and the DSCL.

The following notations have been used in the present document:

- The notation <resourceName> means a placeholder for an identifier of a resource of a certain type. The actual name of the resource is not predetermined.
- The notation "attribute" denotes a placeholder for one or more fixed names. Attribute names and types are described in a table for each resource showing the resource structure.
- Without the delimiters < and > or "and", names appearing in boxes are literals for fixed resource names or attributes.
- Square boxes are used for resources and sub-resources. In order to be able to access sub-resources in a RESTful way, an Issuer shall access these resources directly, only by their references (URIs) which are part of the parent resource representation. The parent resource does not include the representation of its sub-resources, Any deviations from this rules are described in details through the present document.
- Rounded boxes are used for attributes of resources. In order to be able to address and access individual attributes, or parts of an attribute, in a RESTful way they shall be accessible in a way similar to as sub-resources. This is called partial addressing. The main difference is that the attributes are served as part of the containing resource (e.g. in case of http binding they do not have separate e-tag handling and modification times are not kept per attribute).
- Parent-Child relationship and multiplicity: The parent-child relationships are indicated by solid lines. At each end of a line an indicator for the number of valid elements of a parent/child is depicted. The symbol:
  - "*" indicates any number from 0 to infinity.
  - "k", "n", "m", etc. indicate a fixed but so far undefined number of elements. If a parent resource is deleted, the containment relation implies that all child-resource shall be deleted as well (recursively).

The following conventions are used throughout the present document:

- All resource type names shall be in lower case, in case of composed name the subsequent words shall start with a capitol letter, for example "accessRight".
- All resources shall indicate an object and not an action or verb.
- If a resource identifies a collection,, then the resource name shall be in a plural form, for example if we need to indicate a collection of flowers the correspondent resource shall be called "flowers".
- All attributes shall be in lower case. In case of composed name the separation between word is indicated by the using a capitol letter for the following word. For example "searchStrings".

9.2.3.2 Resource <*sclBase*>

The <*sclBase*> resource in this tree shall be the root for all resources that are residing on the hosting SCL.
This SCL can be reached for registering remote SCLs and/or local Applications. The <*sclBase*> hosts and manages sub-resources. An SCL shall perform M2M procedures upon requests by other entities (other SCLs or Applications). Not all defined features may be supported by specific SCLs. For example; a DSCL or GSCL may not be server capable and/or publicly addressable. This implies that some resources may only be addressed locally and not from a remote SCL. A <*sclBase*>-resource shall be addressed by an URI.

The <*sclBase*> resource may contain attributes that describe the hosting SCL.

Regardless of the accessRight, the <*sclBase*> resource shall not be created or deleted via the RESTful API. The <*sclBase*> is managed outside the scope of the API. However, it may be modified and read via the API by entities that have the correct authorization, as defined by the accessRight identified by the accessRightID attribute in the <*sclBase*> resource.

The <*sclBase*> resource does contain collection resources representing collections of <*scl*> resources, <*application*> resources, <*container*> resources, <*group*> resources, <*accessRight*> resources and <*subscription*> resources. In general, where in the resource tree an entity (Application or SCL) creates a <*container*>, a <*group*> or an <*accessRight*> resource depends *mainly* on the lifecycle requirements on that resource.

Resources created directly in a child collection of <*sclBase*> resource live as long as the <*sclBase*> lives, and this gives the resource the possibility to outlive its creator. The creator can either be a local or remote to the <*sclBase*>.

Resources created in the child collection of a <*scl*> resource on a remote <*sclBase*> resource live as long as the <*scl*> resource is available. They will be removed at deregistration of the SCL.

Resources created in the child collection of the <*application*> resource or a local application will live as long as the local <*application*> resource is available. They will be removed at deregistration of the application.

Resources created in the child collection of the <*applicationAnnc*> resource will live as long as that announcement resource lives. They will be removed when de-announcing of the application.

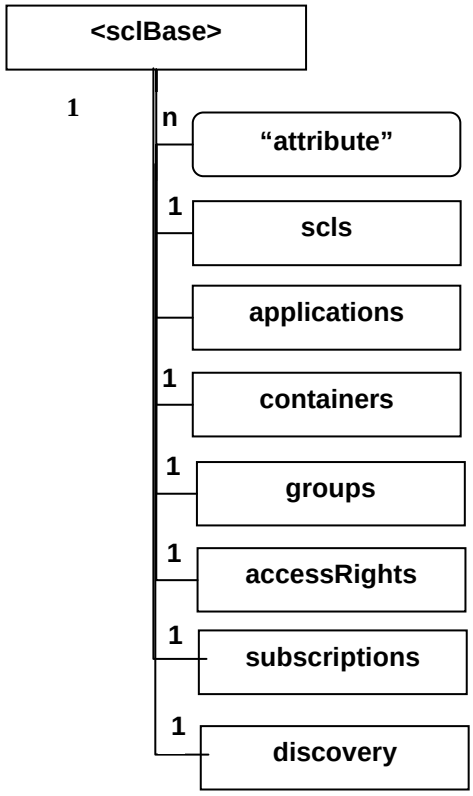


Figure 9.2: Structure of <sclBase>-resources

The <*sclBase*> shall contain the sub resources based on the multiplicity indicated in Table 9.3.

Table 9.3

subResource	Multiplicity	Description
scls	1	Collection of <*scl*> resources each representing a remote SCLs with which the hosting SCL is registered to or that is registered with the hosting SCL. The collection only contains <*scl*> resources, representing remote SCLs. See clause 9.2.3.3.
applications	1	Collection of <*application*> resources which are registered the hosting SCL represented by the <*sclBase*> resource. This collection contains only <application> resources, representing local Applications. See clause 9.2.3.5.
containers	1	Collection of <*container*> resources that do not have a containment relation with a specific remote entity (Application or SCL). This means that if the entity that created a <*container*> in this collection is deleted, the <*container*> shall not be deleted. This collection contains local <*container*> resources (representing local containers created by local or remote entities). See clause 9.2.3.11.
groups	1	Collection of <*group*> resources that do not have a containment relation with a specific remote entity (Application or SCL). This means that if the entity that created a <*group*> in this collection is deleted, the <*group*> resource shall not be deleted. This collection contains local <*group*> resources (representing local groups created by local or remote entities). See clause 9.2.3.18.
accessRights	1	Collection of <*accessRight*> resources that do not have a containment relation with a specific remote entity (Application or SCL). This means that if the entity that created an <*accessRight*> in this collection is deleted, the <*accessRight*> shall not be deleted. This collection contains local <*accessRight*> resources created by local or remote entities. See clause 9.2.3.8.
subscriptions	1	Collection containing all active subscriptions for the <*sclBase*> resource. See clause 9.2.3.22.
discovery	1	Resource used for resource discovery. See clause 9.2.3.36.

The <*sclBase*> shall contain the attributes that are tagged M (Mandatory) in Table 9.4. The <*sclBase*> resource may also contain attributes that are tagged O (Optional) in Table 9.4.

Table 9.4

AttributeName	Mandatory/Optional	Type	Description
accessRightID	O	RW	See clause 9.2.2 Common attributes. The default may be set by the system at creation.
searchStrings	M	RW	See clause 9.2.2 Common attributes. This attribute is only applicable on the registered-to SCL's <sclBase> resource. These searchStrings can be used by the registering SCL when creating a <scl> resource representing the registered-to SCL. This allows the registered-to SCL to be discovered using search strings when the discovery procedure is executed on the discovery resource in the registering SCL.
creationTime	M	RO	See clause 9.2.2 Common attributes.
lastModifiedTime	M	RO	See clause 9.2.2 Common attributes.

AttributeName	Mandatory/Optional	Type	Description
aPocHandling	O	RW	The *aPocHandling* attribute controls how SCL retargeting shall be performed. It can have two value; SHALLOW or DEEP. SHALLOW means that only exact or shallow prefix matches (1 level deep) to elements in the *aPoCPaths* attribute are retargeted, DEEP means that any prefix match will result in retargeting. See clause 9.3.2.31 for details If the *aPocHandling* attribute is not present, the SCL shall act the same as if the value was SHALLOW. This attribute shall only be modified as part of the scl registration procedure (see [create <scl>) or, in case sclBase resource represents a GSCL or DSCL, it can be modified by an NSCL with which the GSCL or DSCL is registered.

9.2.3.3 Resource *scls*

The *scls* resource is a collection resource that shall represent a collection of 0 or more <*scl*> resources.

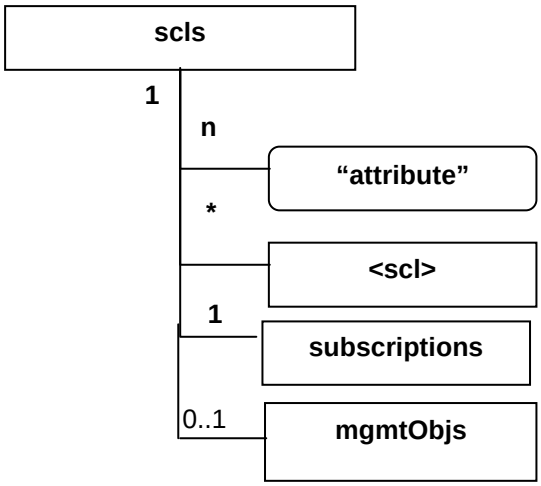


Figure 9.3: Structure of *scls* resource

The *scls* resource shall contain the sub-resources with the indicated multiplicity in Table 9.5.

Table 9.5

subResource	Multiplicity	Description
<scl>	0..unbounded	See clause 9.2.3.4.
subscriptions	1	See clause 9.2.3.22.
mgmtObjs	0..1	This resource is only applicable and mandatory in the NSCL, i.e.: In the NSCL the multiplicity shall be 1. In the DSCL and GSCL, the multiplicity shall be 0.

The *scls* resource shall contain the sub resources that are tagged M (Mandatory) in Table 9.6. The *scls* resource may also contain sub resources that are tagged O (Optional) in Table 9.6.

Table 9.6

AttributeName	Mandatory/Optional	Type	Description
accessRightID	O	RW	See clause 9.2.2 Common attributes
creationTime	M	RO	See clause 9.2.2 Common attributes
lastModifiedTime	M	RO	See clause 9.2.2 Common attributes

9.2.3.4 Resource <*scl*>

An <*scl*> resource shall represent a remote SCL that is registered to the containing <*sclBase*>. This means that each remote SCL that is registered with the <*sclBase*> shall be represented by an <*scl*> resource in that <*sclBase*> (the registered remote SCL).

Conversely, each registered to SCL shall also be represented as a sub-set <*scl*> resource in the registering SCL's <*sclBase*>.

For example, when SCL1 registers with SCL2, there will be two <*scl*> resources created, one in SCL1, <*sclBase1*>/*scls*/<*scl2*> and one in SCL2: <*sclBase2*>/*scls*/<*scl1*>.

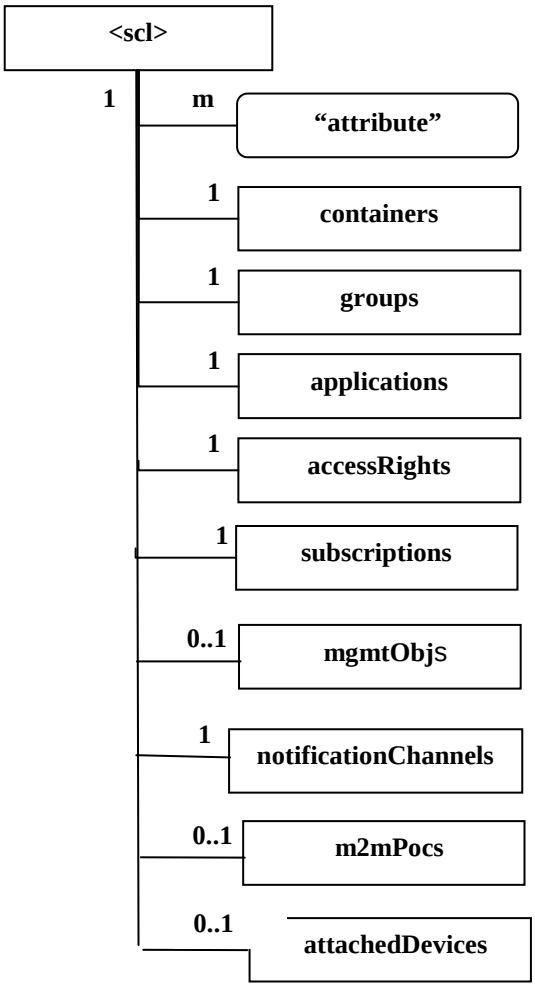


Figure 9.4: Structure of <*scl*> resource

The <*scl*> resource shall contain the sub-resources with the indicated multiplicity in Table 9.7.

Table 9.7

subResource	Multiplicity	Description
containers	1	See clause 9.2.3.11. Can contain a mix of local <*container*> resources created by an SCL or Application and <*containerAnnc*> resources (representing references to remote containers that are announced here).
groups	1	See clause 9.2.3.18. Can contain a mix of local <*group*> resources created by an SCL or Application and <*groupAnnc*> resources (representing references to remote groups that are announced here).
applications	1	See clause 9.2.3.5. Contains only <*applicationAnnc*> resources (representing remote Applications that are announced here).
accessRights	1	See clause 9.2.3.8. Can contain a mix of local <*accessRight*> resources and <*accessRightAnnc*> resources (representing references to remote <*accessRight*> resources that are announced here).
subscriptions	1	See clause 9.2.3.22.
mgmtObjs	0..1	See clause 9.2.3.26. Can contain the <*mgmtObj*> resources. This is only applicable for <*scl*> resource hosted on the NSCL.
notificationChannels	1	See clause 9.2.3.34. The collection of notification channels for use by the registered <*scl*>.
m2mPocs	0..1	See clause 9.2.3.24. Collection of zero or more point of contacts that can be used to reach the SCL for M2M REST traffic, i.e. in-band points of contact. Note that the m2mPoc does not play any role in the communication from registering SCL to registered-to SCL. It is assumed that registering SCL is configured or has discovered a remote/registered-to SCL and in doing so became aware of its IP-address or FQDN. This address will be used for all communication from registering/registered to register SCL. However, for the register SCL to contact the registered SCL, the m2mPocs information will be used. Only the applicable in the NSCL's <*sclBase*>.
attachedDevices	0..1	See clause 9.2.3.32. Collection of zero or more <attachedDevice> resources of the D' (or d) type devices attached to the M2M Gateway (or D device) represented by the <*scl*>. This sub-resource shall only appear in the G (or D) <*scl*> registered in an NSCL resource tree.

The <*scl*> resource shall contain the attributes that are tagged M (Mandatory) in Table 9.8. The <*scl*> resource may also contain attributes that are tagged O (Optional) in Table 9.8.

Table 9.8

AttributeName	Mandatory/Optional	Type	Description
pocs	M	RW	List of zero or more points of contact that can be used for out-of-band communication with an SCL. This is only applicable for <*scl*> resources hosted on the NSCL, i.e. for <*scl*> resources that represent a registered DSCL or GSCL. For <*scl*> resource hosted on the DSCL or GSCL, and therefore representing the NSCL, the value of the pocs is always the empty collection.
remTriggerAddr	O	RW	Contains the "triggering address" of the remote entity represented by the M2M Device's or Gateway's <*scl*> registered resource with a hosting network <*sclBase*> for management purpose. It shall not be present in any registered network <*scl*> hosted in a DSCL or GSCL. This attribute may be provided during the <*scl*> registration procedure or by other provisioning means including pre-configuration or bootstrapping. The format of this attribute shall be a URI as specified by existing management protocols (e.g. a WAP/SIP/HTTP Push URI in OMA-DM, a HTTP URI in BBF TR-069). For network-initiated management procedures, the NSCL shall send a triggering message (e.g. OMA-DM Notification, BBF TR-069 Connection Request) to this address to request the remote entity to setup a management session with the NSCL.
onlineStatus	M	RO	Indicates if the SCL is reachable for M2M REST traffic. For <*scl*> resource hosted on the NSCL, the value is set as described below. The status is set by the hosting SCL based on the provided m2mPoc information and/or long polling activity. If the <*scl*> resource contains at least one active (online) m2mPoc, then the onlineStatus of that SCL resource shall be set to ONLINE. If the <*scl*> resource is currently involved in long-polling, the online status of that SCL resource shall be set to be ONLINE. If there are no m2mPocs defined or if all m2mPocs are marked as OFFLINE and no long-polling is ongoing, then the onlineStatus of that SCL shall be set to OFFLINE. If there are m2mPocs, and all of them are marked as NOT_REACHABLE, the onlineStatus of the SCL shall be set to NOT_REACHABLE to indicate that the SCL cannot be reached using any of the m2mPocs. NOT_REACHABLE can be regarded as a sub-state of ONLINE. For an <scl> resource hosted on a G/DSCL the value shall be ONLINE
serverCapability	M	RO	When set to TRUE it means that this SCL could try to issue connections towards the registered SCL. This attribute is always set to TRUE for <*scl*> resources hosted on the DSCL or GSCL, i.e. for <*scl*> resources representing an NSCL (i.e. the registered-to SCL). For <*scl*> resource hosted on the NSCL, the value of serverCapability is set to TRUE only if there are m2mPocs available.
link	M	WO	The URI of the <*sclBase*> of the remote SCL.
schedule	O	RW	Represents the connection schedule of the remote / registered SCL. This is provided for information purposes only. The present document does not mandate any specific handling associated with the schedule. Only applicable for registered SCLs, i.e. only in the NSCL's <*sclBase*> resource tree.
expirationTime	M	RW	See clause 9.2.2 Common attributes. This represents the expiration time of the registration. If the SCL does not refresh its registration before that time the resource is deleted.

AttributeName	Mandatory/Optional	Type	Description
accessRightID	O	RW	See clause 9.2.2 Common attributes.
searchStrings	M	RW	See clause 9.2.2 Common attributes.
creationTime	M	RO	See clause 9.2.2 Common attributes.
lastModifiedTime	M	RO	See clause 9.2.2 Common attributes.
locTargetDevice	O	RW	The device address to be used for retrieving the location information of the M2M Device or Gateway which is represented by this <*scl*>. Only present for <*scl*> resource hosted on the NSCL. This attribute is only used in the case that the location information is provided by a network-based location server (e.g. a 3GPP location server). It will be provided to the location server by the hosting SCL (i.e. NTOE) for the location information retrieval. The format of this attributed shall conform to the interface provided by the location server (e.g. MSISDN for a 3GPP location server).
mgmtProtocolType	O	RW	It is defined and used to store the management protocol that this <*scl*> supports. Only applicable for <*scl*> resource hosted on the NSCL. An M2M Device (or a M2M Gateway) shall indicate the mgmtProtocolType they support during the procedures such as SCL Registration, Update SCL Registration, and normal RESTful Update of this attribute.
integrityValResults	O	RW	Indicates the signed Integrity Validation results for the registering SCL. This attribute is optional and relates only to D/GSCL since Integrity Validation is not performed on the NSCL.
aPocHandling	O	RO	The *aPocHandling* attribute as received during scl registration is the basis for *aPocHandling* attribute that is set on the <*sclBase*> resource, which in turn controls the SCL retargeting behaviour.

The *link* attribute is used for routing purposes in the NSCL. If a URI is addressed which is a subordinate URI of the one of the links in a registered SCL, then the hosting SCL (i.e. the NSCL) shall route the request based on the m2mPoc information provided for the registered SCL

The algorithm for this is follows:

- The hosting SCL shall select which of the <*m2mPoc*> resources marked as ONLINE or NOT_REACHABLE shall be used. The selection logic could depend on the indicated RCAT (see clause 9.3.1.5), time of day, cost criteria, location, etc. Not reachable m2mPocs typically would get a lower priority in this selection process. The present document does not specify how the selection of the m2mPoc is done.

- The contactInformation from the selected <*m2mPoc*> resource shall be used to obtain a host and port. The contactInformation either directly contains the host and port or contains information that can be used to derive the host and port information.

- The request is routed based on The resolved host and port.

- If the hosting SCL detects that the constructed URI is not reachable, the hosting SCL shall set the onlineStatus of that selected <*m2mPoc*> resource to NOT_REACHABLE. This might lead to a change of the onlineStatus of the <*scl*> resource as described above. The hosting SCL shall then repeat the process using the remainder of the <*m2mPoc*> resources.

If all of the <*m2mPoc*> resources are OFFLINE, if none of them could be reached, or if the *m2mPocs* collection is empty, then the hosting SCL shall buffer the request for a maximum duration as indicated by the TRPDT (see clause 9.3.1.5). In the mean time it may try to wakeup the SCL that is corresponds to the <*scl*> resource. If no connection becomes available before the TRPDT expires, then the hosting SCL shall send an error response to the request.

9.2.3.5 Resource applications

The *applications* resource shall represent a collection of <*application*> resources.

Depending on its location in the tree it may contain registered application or announced application resources.

The <*sclBase*>*/applications* resource shall contain only <*application*> resources, representing locally registered applications.

The <*sclBase*>*/scls/*<*scl*>*/applications* resource shall contain only <*applicationAnnc*> resources, representing remote applications residing on the indicated <*scl*> that are announced to the local <*sclBase*>.

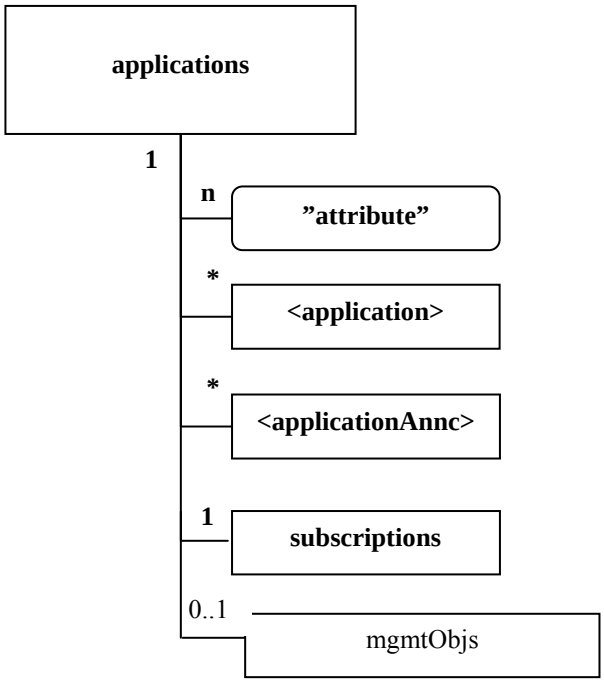


Figure 9.5: Structure of registered applications and applications resources

The *applications* resource shall contain the sub with the indicated multiplicity in Table 9.9.

Table 9.9

subResource	Multiplicity	Description
<application>	0..unbounded	See clause 9.2.3.6. Represents either a locally registered or registerable applications.
<applicationAnnc>	0..unbounded	See clause 9.2.3.7. Represents remote applications that are announced here.
subscriptions	1	See clause 9.2.3.22.
mgmtObjs	0..1	If the *applications* resource resides under <*nsclBase*>*/scls/*<*scl*>,i.e. on the NSCL of the M2M Server (NREM), it shall have a collection *mgmtObjs*. If *applications* resource resides under <*sclBase*>*/scls/*<*scl*> it shall not have *mgmtObjs*. When the *applications* resource resided in a M2M Device (DREM) or M2m Gateway (GREM), it will not have *mgmtObjs*.

The *applications* resource shall contain the attributes that are tagged M (Mandatory) in Table 9.10. The *applications* resource may also contain attributes that are tagged O (Optional) in Table 9.10.

Table 9.10

AttributeName	Mandatory/Optional	Type	Description
accessRightID	O	RW	See clause 9.2.2 Common attributes.
creationTime	M	RO	See clause 9.2.2 Common attributes.
lastModifiedTime	M	RO	See clause 9.2.2 Common attributes.

9.2.3.6 Resource <application>

This resource represents an application that is registered.

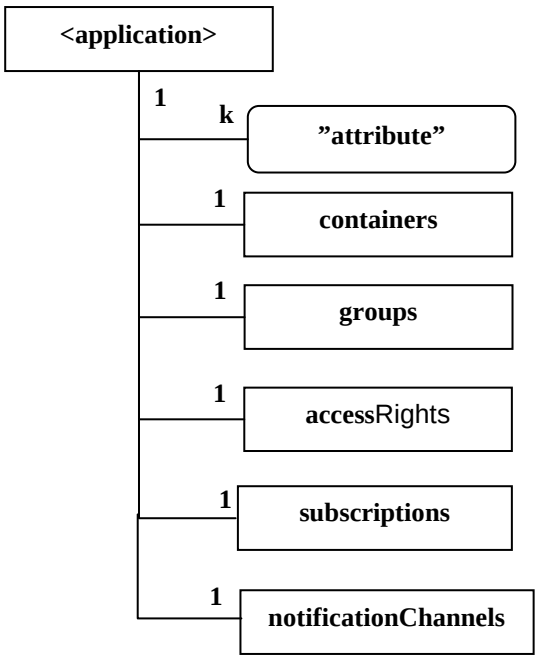


Figure 9.6: Structure of <*application*> resource

The <*application*> resource shall contain the sub resources with the indicated multiplicity in Table 9.11.

Table 9.11

subResource	Multiplicity	Description
containers	1	See clause 9.2.3.11. Contains local containers typically created by the parent application.
groups	1	See clause 9.2.3.18. Contains local groups typically created by the parent application.
accessRights	1	See clause 9.2.3.8. Contains local <*accessRight*> resources typically created by the parent application.
subscriptions	1	See clause 9.2.3.22.
notificationChannels	1	See clause 9.2.3.34. The collection of notification channels for use by the registered application.

The <*application*> resource shall contain the attributes that are tagged M (Mandatory) in Table 9.12. The <*application*> resource may also contain attributes that are tagged O (Optional) in Table 9.12.

Table 9.12

AttributeName	Mandatory/Optional	Type	Description
expirationTime	M	RW	See clause 9.2.2 Common attributes. This shall represent the expiration time of the registration. If the SCL does not refresh its registration before that time the resource is deleted and the application is de-registered.
accessRightID	O	RW	See clause 9.2.2 Common attributes.
searchStrings	M	RW	See clause 9.2.2 Common attributes.
creationTime	M	RO	See clause 9.2.2 Common attributes.
lastModifiedTime	M	RO	See clause 9.2.2 Common attributes.
announceTo	M	RW	See clause 9.2.2 Common attributes.
aPoC	O	RW	The Application Point of Contract is a URI that identifies how request are re-targeted.
aPoCPaths	O	RW	The *aPocPaths*, if present, is used to determine if a targetURI is to be re-targeted, by doing a prefix match against the elements in the path. Each path can optionally have an *accessRightID* associated with it, which, if present, is used for authorization purposes when doing the re-targeting. The *accessRightID* of the best matching path prefix is used for this purpose. The value of *aPoCPaths* is only relevant when the *aPoC* attribute is also present.
locRequestor	O	RW	The identity of the Application to be used for the content of privacy control when requesting the location information of a remote M2M Device or Gateway. This attribute is only used in the case that the location information is provided by a network-based location server (e.g. a 3GPP location server). It will be provided to the location server if the content is required for the location information retrieval. The format of this attributed shall conform to the interface provided by the location server (e.g. MSISDN for a 3GPP location server).

9.2.3.7 Resource <applicationAnnc>

This resource shall represent an active announcement of a registered application in another SCL. The <*applicationAnnc*> resource keeps a link to the original resource. That will also be the reference returned during discovery.

The <*applicationAnnc*> resources contains sub-resources for *containers*, *groups*, and *accessRights* collection resources. This allow the creation of these types of resources (container/group/accessRight) with a lifetime and scope that are linked to the announced resource. I.e. resource created as descendants of the announced resource will automatically be deleted when the application is de-announced or when the announcement expires.

It shall be the responsibility of the announcing SCL to keep the accessRightID and the searchStrings in sync with the resource that is announced.

The following rules apply:

<*sclBase1./applications/*<*app*> announced on <*sclBase2*> as <*sclBase2*>*/scls/*<*scl1*>*/applications/*<*app_Annc*>

Where <*scl1*> is the resource created when <*sclBase1*> registered with <*sclBase1*>. The announcing SCL shall suggest the name of the announced resource as the concatenation of the name of the announcing resource with the suffix "Annc" i.e. <app>Annc. Or if the application resource name is "myApp", the announced resource shall be named "myAppAnnc".

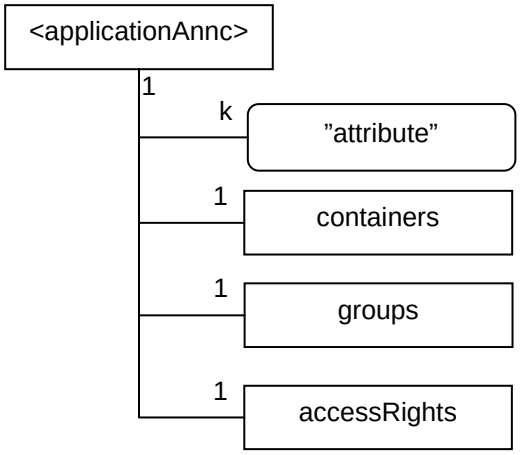


Figure 9.7: Structure of <*applicationAnnc*> resource

The <*applicationAnnc*> resource shall contain the sub with the indicated multiplicity in Table 9.13.

Table 9.13

subResource	Multiplicity	Description
containers	1	See clause 9.2.3.11. Contains a mix or containerAnnc and container resources.
groups	1	See clause 9.2.3.18. Contains a mix of groupAnnc and group resources.
accessRights	1	See clause 9.2.3.8. Contains a mix of accessRightAnnc and accessRight resources.

The <*applicationAnnc*> resource shall contain the attributes that are tagged M (Mandatory) in Table 9.14. The <*applicationAnnc*> resource may also contain attributes that are tagged O (Optional) in Table 9.14.

Table 9.14

AttributeName	Mandatory/Optional	Type	Description
link	M	WO	See clause 9.2.2 Common attributes. The reference to the resource that is announced here.
accessRightID	O	RW	See clause 9.2.2 Common attributes. Same as the *accessRightID* of the announced resource (indicated by the link).
searchStrings	M	RW	See clause 9.2.2 Common attributes. Same as the searchStrings of the announced resource (indicated by the link).
expirationTime	M	RW	See clause 9.2.2 Common attributes.

9.2.3.8 Resource accessRights

The *accessRights* resource represents a collection of <*accessRight*> resources and/or <*accessRightAnnc*> resources. The following combinations are possible:

<*sclBase*>*/accessRights* - contains accessRight resources only (created by local or remote entities).

<*sclBase*>*/scls/*<*scl*>*/accessRights* - contains accessRightAnnc resources announced by <*scl*> and/or accessRight resources.

<*sclBase*>*/applications/*<*app*>*/accessRights* - contains local accessRight resource only, typically created by the Application corresponding to <app>.

<sclBase>/scls/<scl>/applications/<applicationAnnc>/accessRights - contains accessRightAnnc resource announced by the SCL corresponding to *<scl>* and/or accessRight resource typically created by the SCL corresponding to *<scl>* or the Application on whose behalf *<applicationAnnc>* is created.

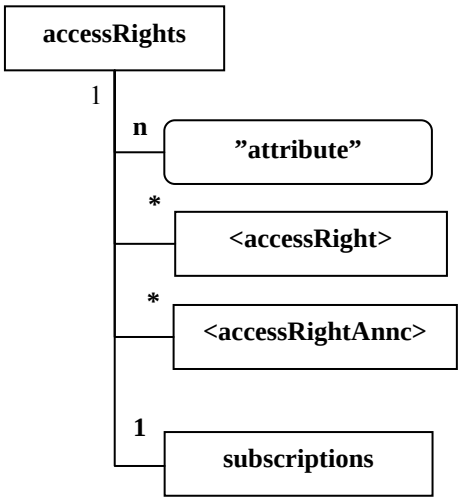


Figure 9.8: Structure of accessRights-resource

The accessRights resource shall contain the sub resources with the indicated multiplicity in Table 9.15.

Table 9.15

subResource	Multiplicity	Description
<accessRight>	0..unbounded	See clause 9.2.3.9.
<accessRightAnnc>	0..unbounded	See [accessRightAnnc].
subscriptions	1	See clause 9.2.3.22.

The accessRights resource shall contain the attributes that are tagged M (Mandatory) in Table 9.16. The accessRights resource may also contain attributes that are tagged O (Optional) in Table 9.16.

Table 9.16

AttributeName	Mandatory/Optional	Type	Description
accessRightID	O	RW	See clause 9.2.2 Common attributes.
creationTime	M	RO	See clause 9.2.2 Common attributes.
lastModifedTime	M	RO	See clause 9.2.2 Common attributes.

9.2.3.9 Resource <accessRight>

Access rights are defined as "white lists" or permissions, i.e. each permission defines "allowed" entities (defined in the permissionHolders) for certain access modes (permissionFlags). Sets of permissions are handled such that the resulting permissions for a group of permissions are the sum of the individual permissions. I.e. an action is permitted if it is permitted by some / any permission in the set.

By setting an accessRightID attribute on a resource, the permissions for accessing that resource are then defined by the permissions defined in the accessRight resource.

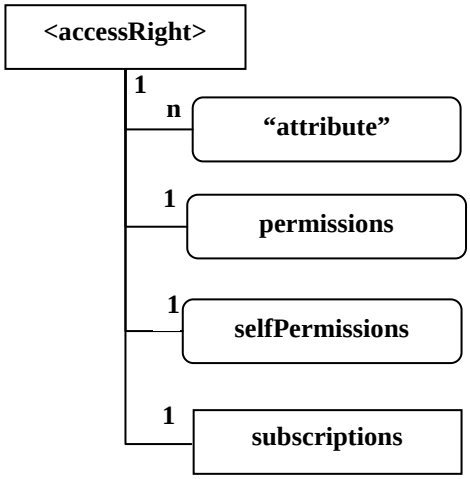


Figure 9.9: Structure of <*accessRight*> resource

The <*accessRight*> resource shall contain the sub resources with the indicated multiplicity in Table 9.17.

Table 9.17

subResource	Multiplicity	Description
subscriptions	1	See clause 9.2.3.22.

The <*accessRight*> resource shall contain the attributes that are tagged M (Mandatory) in Table 9.18. The <*accessRight*> resource may also contain attributes that are tagged O (Optional) in Table 9.18.

Table 9.18

AttributeName	Mandatory/Optional	Type	Description
expirationTime	M	RW	See clause 9.2.2 Common attributes.
searchStrings	M	RW	See clause 9.2.2 Common attributes.
creationTime	M	RO	See clause 9.2.2 Common attributes.
lastModifiedTime	M	RO	See clause 9.2.2 Common attributes.
announceTo	M	RW	See clause 9.2.2 Common attributes.
permissions	M	RW	The collection of permissions defined by this <*accessRight*>. These permissions are applied to resources referencing this accessRight resource using the *accessRightID* attribute.
selfPermissions	M	RW	Defines the collection of permissions for the <*accessRight*> resource itself.

The permissionFlags and the permissionHolders could then be generalized to actions (which might be granting access, but might also be more specific, like granting access to a subset, i.e. filtering part of the data). The permissionHolders could be generalized to conditions, which may include things like the identity of the requestor, everybody except specified identities, but it might also include time based conditions, etc.

9.2.3.10 Resource <accessRightAnnc>

This resource shall represent an active announcement of an <*accessRight*> resource in another SCL. The <*accessRightAnnc*> resource keeps a link to the original resource. That will also be the reference returned during discovery.

It shall be the responsibility of the announcer to keep the *accessRightID* and the *searchStrings* attributes in sync with the resource that is announced.

The following rules apply:

<sclBase1>/accessRights/<accessRight> announced on *<sclBase2>* as *<sclBase2>/scls/<scl1>/accessRights/<accessRightAnnc>*.

<sclBase1>/applications/<app>/accessRights/<accessRight> announced on *<sclBase2>* as *<sclBase2>/scls/<scl1>/applications/<appAnnc>/accessRights/<accessRightAnnc>*.

Where *<scl1>* is the resource created when *<sclBase1>* registered with *<sclBase1>* and *<appAnnc>* is the announced resource for *<app>*.

The announcing SCL shall suggest the name of the announced resource as the concatenation of the name of the announcing resource with the suffix "Annc", i.e. *<accessRight>Annc*. Or if the *<accessRight>* resource name is "myAccessRight", the announced resource shall be named "myAccessRightAnnc". The hosting/announced-to SCL may not accept this name in case of name clashes with locally created accessRight resource. In such a case the announced-to SCL will allocate a different name and return this to the issuer.

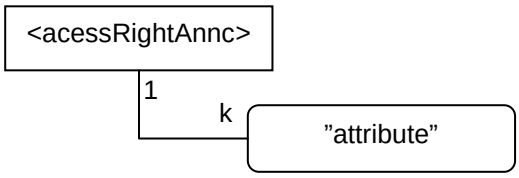


Figure 9.10: Structure of *<accessRightAnnc>* resource

The *<accessRightAnnc>* resource shall not contain the any sub resources:

The *<accessRightAnnc>* resource shall contain the attributes that are tagged M (Mandatory) in Table 9.19. The *<accessRightAnnc>* resource may also contain attributes that are tagged O (Optional) in Table 9.19.

Table 9.19

AttributeName	Mandatory/Optional	Type	Description
link	M	WO	See clause 9.2.2 Common attributes. The reference to the resource that is announced here.
accessRightID	O	RW	See clause 9.2.2 Common attributes. Same as the *accessRightID* of the announced resource (indicated by the link).
searchStrings	M	RW	See clause 9.2.2 Common attributes. Same as the searchStrings of the announced resource (indicated by the link).
expirationTime	M	RW	See clause 9.2.2 Common attributes.

9.2.3.11 Resource containers

The *containers* resource represents a collection of container resources and containerAnnc resources. The following combinations are possible:

<sclBase>/containers – can contain the following type of resources:

- container resources only (either created by local or remote entities)

<sclBase>/scls/<scl>/containers – can contain a mix of the following type of resources:

- containerAnnc resources announced by the SCL corresponding to *<scl>*
- container resources

<sclBase>/applications/<app>/containers – can contain a mix of the following type of resources:

- container resources, typically created by the Application corresponding to <app>

- locationContainer, typically created by the Application corresponding to <app>

<sclBase>/scls/<scl>/applications/<applicationAnnc>/containers – can contain a mix of the following type of resources:

- <*containerAnnc*> resources announced by the SCL corresponding to <*scl*>
- <*locationContainerAnnc*> resources announced by the SCL corresponding to <*scl*>
- <*container*> resources typically created by the SCL corresponding to <*scl*> or corresponding to the Application on whose behalf <*applicationAnnc*> is created
- <*locationContainer*> resources typically created by the SCL corresponding to <*scl*> or corresponding to the Application on whose behalf <*applicationAnnc*> is created

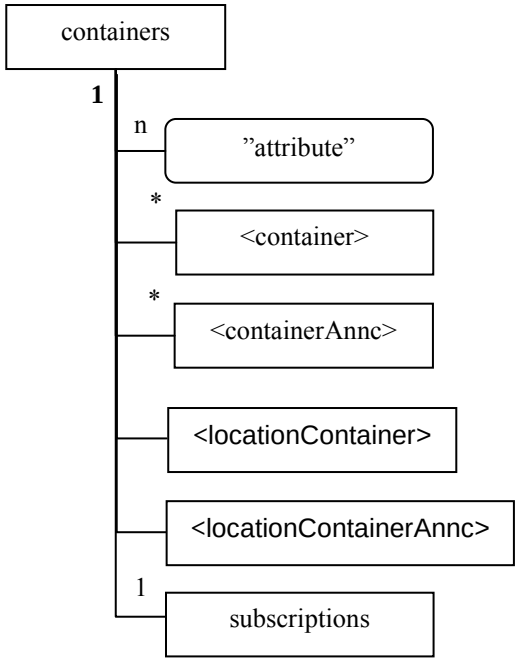


Figure 9.11: Structure of containers-resource

The *containers* resource shall contain the sub resources that are tagged M (Mandatory) with the indicated multiplicity in Table 9.20. The *containers* resource may also contain sub resources that are tagged O (Optional) with the indicated multiplicity in Table 9.20.

Table 9.20

subResource	Mandatory/Optional	Multiplicity	Description
<container>	M	0..unbounded	See clause 9.2.3.12.
<containerAnnc>	M	0..unbounded	See clause 9.2.3.13.
<locationContainer>	O	0..unbounded	See clause 9.2.3.14. <*locationContainer*> shall only reside under the *containers* resource, of which the parent resource is <*application*> or <*applicationAnnc*> resource.
<locationContainerAnnc>	O	0..unbounded	See clause 9.2.3.15. <*locationContainerAnnc*> shall only reside under the *containers* resource, of which the parent resource is <*application*> or <*applicationAnnc*> resource.
subscriptions	M	1	See clause 9.2.3.22.

The *containers* resource shall contain the attributes that are tagged M (Mandatory) in Table 9.21. The *containers* resource may also contain attributes that are tagged O (Optional) in Table 9.21.

Table 9.21

AttributeName	Mandatory/Optional	Type	Description
accessRightID	O	RW	See clause 9.2.2 Common attributes.
creationTime	M	RO	See clause 9.2.2 Common attributes.
lastModifiedTime	M	RO	See clause 9.2.2 Common attributes.

9.2.3.12 Resource <container>

The <*container*> resource represents a container for instances.

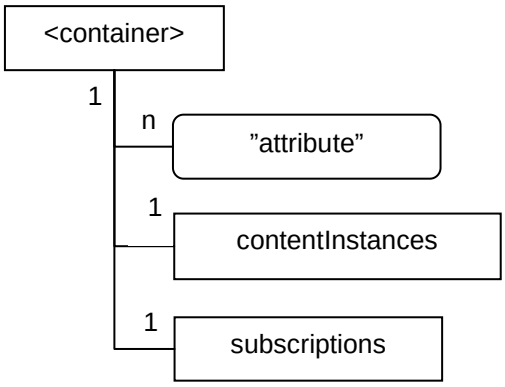


Figure 9.12: Structure of <*container*> resource

The <*container*> resource shall contain the sub resources with the indicated multiplicity in Table 9.22.

Table 9.22

subResource	Mandatory/Optional	Multiplicity	Description
contentInstances	M	1	See clause 9.2.3.16. Collection of 0 or more instances.
subscriptions	M	1	See clause 9.2.3.22.

The <*container*> resource shall contain the attributes that are tagged M (Mandatory) in Table 9.23. The <*container*> resource may also contain attributes that are tagged O (Optional) in Table 9.23.

Table 9.23

AttributeName	Mandatory/Optional	Type	Description
expirationTime	M	RW	See clause 9.2.2 Common attributes.
accessRightID	O	RW	See clause 9.2.2 Common attributes.
searchStrings	M	RW	See clause 9.2.2 Common attributes.
creationTime	M	RO	See clause 9.2.2 Common attributes.
lastModifiedTime	M	RO	See clause 9.2.2 Common attributes.
announceTo	M	RW	See clause 9.2.2 Common attributes.
maxNrOfInstances	M	RW	Maximum number of instances of a <*container*> resource.
maxByteSize	M	RW	Maximum number of bytes that is allocated for a <*container*> resource for the overall instances.
maxInstanceAge	M	RW	Maximum age of instances of a <*container*> resource, the value is expressed in seconds.

9.2.3.13 Resource <containerAnnc>

This resource shall represent an active announcement of a container in another SCL. The containerAnnc resource keeps a link to the original resource. That will also be the reference returned during discovery.

It shall be the responsibility of the announcer to keep the *accessRightID* and the *searchStrings* attribute in sync with the resource that is announced.

The following rules apply:

<*sclBase1*>/*containers*/<*container*> announced on <*sclBase2*> as
<*sclBase2*>/*scls*/<*scl1*>/*containers*/<*container*>*Annc*.

<*sclBase1*>/*applications*/<*app*>/*containers*/<*container*> announced on <*sclBase2*> as
<*sclBase2*>/*scls*/<*scl1*>/*applications*/<*appAnnc*>/*containers*/<*containerAnnc*>

Where <*scl1*> is the resource created when <*sclBase1*> registered with <*sclBase1*> and <*appAnnc*> is the announced resource for <*app*>.

The announcing SCL shall suggest the name of the announced resource as the concatenation of the name of the announcing resource with the suffix "Annc", i.e. <*container*>*Annc*. Or if the <*container*> resource name is "myContainer", the announced resource shall be named "myContainerAnnc". The hosting/announced-to SCL may not accept this name in case of name clashes with locally created <*container*> resource. In such a case the announced-to SCL will allocate a different name and return this to the issuer.

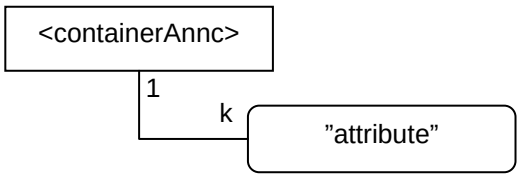


Figure 9.13: Structure of <*containerAnnc*> resource

The <*containerAnnc*> resource shall not contain the any sub resources:

The <*containerAnnc*> resource shall contain the attributes that are tagged M (Mandatory) in Table 9.24. The <*containerAnnc*> resource may also contain attributes that are tagged O (Optional) in Table 9.24.

Table 9.24

AttributeName	Mandatory/Optional	Type	Description
link	M	WO	See clause 9.2.2 Common attributes. The reference to the resource that is announced here.
accessRightID	O	RW	See clause 9.2.2 Common attributes. Same as the *accessRightID* of the announced resource (indicated by the link).
searchStrings	M	RW	See clause 9.2.2 Common attributes. Same as the searchStrings of the announced resource (indicated by the link).
expirationTime	M	RW	See clause 9.2.2 Common attributes.

9.2.3.14 Resource <locationContainer>

The location resource contains location information of a remote entity.

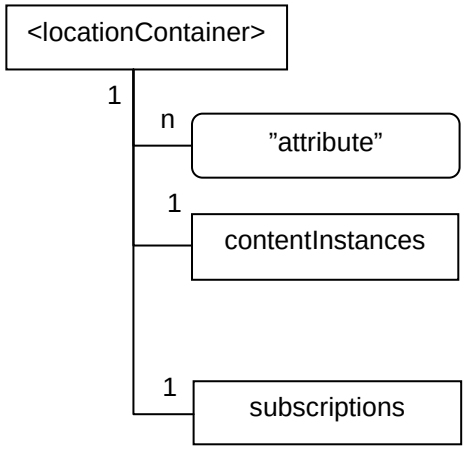


Figure 9.14: Structure of <location> resource

The <*locationContainer*> resource shall contain the sub resources with the indicated multiplicity in Table 9.25.

Table 9.25

subResource	Mandatory/Optional	Multiplicity	Description
contentInstances	M	1	See clause 9.2.3.16 Collection of zero or more instances of location information of a remote entity. The *contentInstances* resource may not actually store the location data instances, if the location information is provided by a location server in the Network Domain.
subscriptions	M	1	See clause 9.2.3.22.

The <*locationContainer*> resource shall contain the attributes that are tagged M (Mandatory) in Table 9.26. The <*locationContainer*> resource may also contain attributes that are tagged O (Optional) in Table 9.26.

Table 9.26

AttributeName	Mandatory/Optional	Type	Description
expirationTime	M	RW	See clause 9.2.2 Common attributes.
accessRightID	O	RW	See clause 9.2.2 Common attributes. Referencing to the accessRight resource which controls the privacy of the location information retrieval.
searchStrings	M	RW	See clause 9.2.2 Common attributes.
creationTime	M	RO	See clause 9.2.2 Common attributes.
lastModifiedTime	M	RO	See clause 9.2.2 Common attributes.
announceTo	M	RW	See clause 9.2.2 Common attributes.
maxNrOfInstances	M	RW	Maximum number of instances of a <*container*> resource.
maxByteSize	M	RW	Maximum number of bytes that is allocated for a <*container*> resource for the overall instances.
maxInstanceAge	M	RW	Maximum age of instances of a <*container*> resource, the value is expressed in seconds.
locationContainerType	M	WO	Indicates the source of the location information, i.e. whether it is application generated or location server based. In the location server based case, a retrieval request addressing the <*locationContainer*> resource shall trigger the hosting SCL to retrieve the corresponding location information from a location server, using the information provided in the "locRequestor" attribute of the requesting <*application*> resource and the "locTargetDevice" attribute of the parent <*scl*> resource of this <*locationContainer*> resource.

9.2.3.15 Resource <locationContainerAnnc>

This resource shall represent an active announcement of a <*locationContainer*> resource in another SCL. The locationContainerAnnc resource keeps a link to the original resource. That will also be the reference returned during discovery.

It shall be the responsibility of the announcer to keep the accessRight and the searchStrings in sync with the resource that is announced.

The following rules apply:

<*sclBase1*>/*applications*/<*app*>/*containers*/<*locationContainer*> announced on <*sclBase2*> as <*sclBase2*>/*scls*/<*scl1*>/*applications*/<*appAnnc*>/*containers*/<*locationContainerAnnc*>.

Where <*scl1*> is the resource created when <*sclBase1*> registered with <*sclBase1*> and <*appAnnc*> is the announced resource for <*app*>.

The announcing SCL shall suggest the name of the announced resource as the concatenation of the name of the announcing resource with the suffix "Annc", i.e. <*locationContainer*>*Annc*. Or if the <*locationContainer*> resource name is "myLocationContainer", the announced resource shall be named "myLocationContainerAnnc". The hosting/announced-to SCL may not accept this name in case of name clashes with locally created <*locationContainer*> resource. In such a case the announced-to SCL will allocate a different name and return this to the issuer.

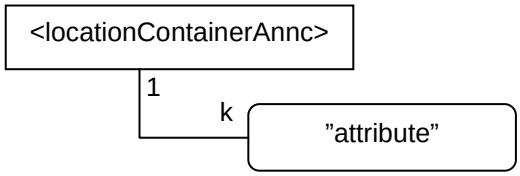


Figure 9.15: Structure of *<locationContainerAnnc>* resource

The <*locationContainerAnnc*> resource shall not contain the any sub resources.

The <*locationContainerAnnc*> resource shall contain the attributes that are tagged M (Mandatory) in Table 9.27. The <*locationContainerAnnc*> resource may also contain attributes that are tagged O (Optional) in Table 9.27.

Table 9.27

AttributeName	Mandatory/Optional	Type	Description
link	M	WO	See clause 9.2.2 Common attributes. The reference to the resource that is announced here.
accessRightID	O	RW	See clause 9.2.2 Common attributes. Same as the *accessRightID* of the announced resource (indicated by the link).
searchStrings	M	RW	See clause 9.2.2 Common attributes. Same as the searchStrings of the announced resource (indicated by the link).

9.2.3.16 Resource contentInstances

The *contentInstances* resource represents the collection of content instances in a container. It shall also keep track of the latest (newest) and the oldest instance. These shall reference the newest instance and the oldest instance in the collection, respectively. If there are no instances in the collection, the latest and oldest resources shall not be present.

The *contentInstances* resource is special in that a retrieve on the *contentInstances* shall give in the returned resource representation the content of the <*contentInstance*> resources in the collection (subject to the filtercriteria) and not just the references to these child resources. When a retrieve is performed on the *contentInstances* resource it shall be possible to indicate that either only the meta-data of the contentInstance resources in the collection matching the filter criteria shall be returned or whether both the meta-data together with the actual content of each contentInstance resource matching the filtercriteria shall be returned.

The *ContentInstances* resource does not have its own AccessRights. Instead the accessRights of the parent <*container*> resource shall apply.

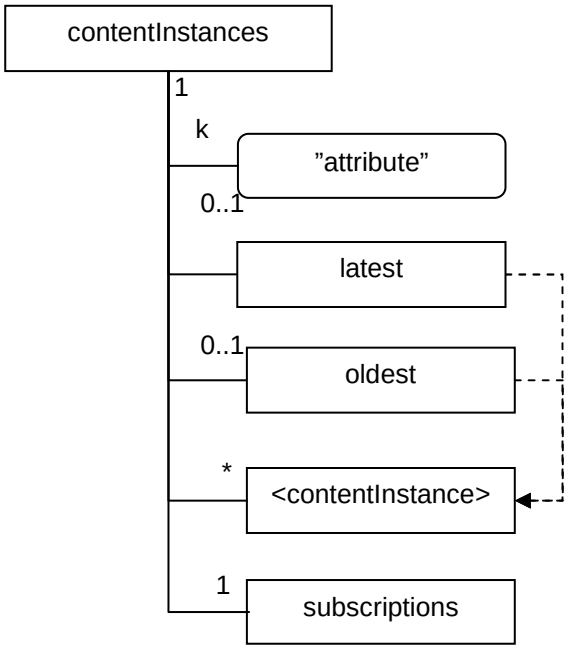


Figure 9.16: Structure of *contentInstances* resource

The *contentInstances* resource shall contain the sub with the indicated multiplicity in Table 9.28.

Table 9.28

subResource	Multiplicity	Description
<contentInstance>	0..unbounded	See clause 9.2.3.17.
latest	0..1	Reference to latest instance, if present.
oldest	0..1	Reference to oldest instance, if present.
subscriptions	1	See clause 9.2.3.22.

The *contentInstances* resource shall contain the attributes that are tagged M (Mandatory) in Table 9.29. The *contentInstances* resource may also contain attributes that are tagged O (Optional) in Table 9.29.

Table 9.29

AttributeName	Mandatory/Optional	Type	Description
creationTime	M	RO	See clause 9.2.2 Common attributes.
lastModifiedTime	M	RO	See clause 9.2.2 Common attributes.
currentNrOfInstances	M	RO	Current number of instances in a <*container*> resource. It is limited by the maxNrOfInstances.
currentByteSize	M	RO	Current size in bytes of data stored in a <*container*> resource. It is limited by the maxNrOfBytes.

9.2.3.17 Resource <contentInstance>

The contentInstance resource represents a data instance in the container. The content of the instance is opaque to the M2M platform and it might even be encrypted. However, there is meta-data associated with an instance which shall be accessible.

Contrary to other resources, the <*contentInstance*> resource cannot be modified once created, regardless of the accessRightID associated with the parent resource. An instance may be deleted explicitly or it may be deleted by the platform based on policies. If the platform has policies for the instance retention these shall be represented by the attributes maxByteSize, maxNrOfInstances and/or maxInstanceAge on the <*container*> resource. If multiple policies are in effect, the strictest policy shall apply.

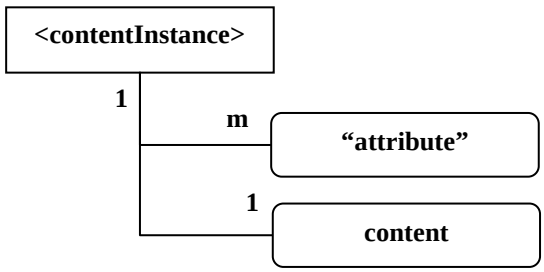


Figure 9.17: Structure of <*contentInstance*> resource

The <*contentInstance*> resource shall contain the attributes that are tagged M (Mandatory) in Table 9.30. The <*contentInstance*> resource may also contain attributes that are tagged O (Optional) in Table 9.30: <*contentInstance*> attributes.

Table 9.30

AttributeName	Mandatory/Optional	Type	Description
contentType	O	WO	Optional type of the content included in the content attribute. This is media-type as defined in [53] and [32].
contentSize	M	WO	Size in bytes of the content attribute.
creationTime	M	RO	See clause 9.2.2 Common attributes.
lastModifiedTime	M	RO	See clause 9.2.2 Common attributes.
delayTolerance	O	WO	The time before the addition of the containing <*contentInstance*> resource shall be notified to any subscribers.
content	M	WO	Real opaque content of an instance. This may for example be an image taken by a security camera, or a temperature measurement taken by a temperature sensor.

9.2.3.18 Resource *groups*

The *groups* resource represents a collection of group resources and/or groupAnnc resources. The following combinations are possible:

<*sclBase*>*/groups* - contains group resources only (created by local or remote entities).

<*sclBase*>*/scls/*<*scl*>*/groups* - contains groupAnnc resources announced by the SCL corresponding to <*scl*> and/or group resources.

<*sclBase*>*/applications/*<*app*>*/groups* - contains local group resource only, typically created by the Application corresponding to <*app*>.

<*sclBase*>*/scls/*<*scl*>*/applications/*<*applicationAnnc*>*/groups* - contains groupAnnc resource announced by the SCL corresponding to <*scl*> and/or group resource typically created by the SCL corresponding to <*scl*> or the Application on whose behalf <*applicationAnnc*> is created.

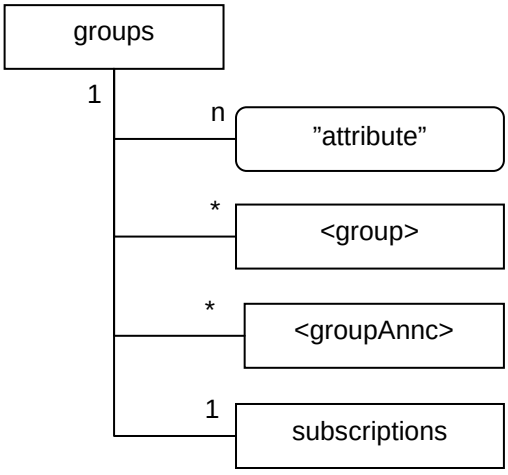


Figure 9.18: Structure of *groups*-resource

The *groups* resource shall contain the sub resources with the indicated multiplicity in Table 9.31.

Table 9.31

subResource	Multiplicity	Description
<group>	0..unbounded	See clause 9.2.3.19.
<groupAnnc>	0..unbounded	See clause 9.2.3.20.
subscriptions	1	See clause 9.2.3.22.

The *groups* resource shall contain the attributes that are tagged M (Mandatory) in Table 9.32. The *groups* resource may also contain attributes that are tagged O (Optional) in Table 9.32.

Table 9.32

AttributeName	Mandatory/Optional	Type	Description
accessRightID	O	RW	See clause 9.2.2 Common attributes.
creationTime	M	RO	See clause 9.2.2 Common attributes.
lastModifiedTime	M	RO	See clause 9.2.2 Common attributes.

9.2.3.19 Resource <*group*>

The group resource represents a group of resources of the same or mixed types. The <*group*> resource can be used to do bulk manipulations on the resources represented by the members. The group contains an attribute that represents the members of the group and a sub-resource (the *membersContent*) that allows verbs to be applied to the resources represented by those members.

When used as one of the permission holders in an accessRight resource, the group can be used to grant a collection of applications (represented by <*application*> resources) or SCLs (represented by <*sclBase*> resources) permissions for accessing (creating children, retrieving, etc.) a resource.

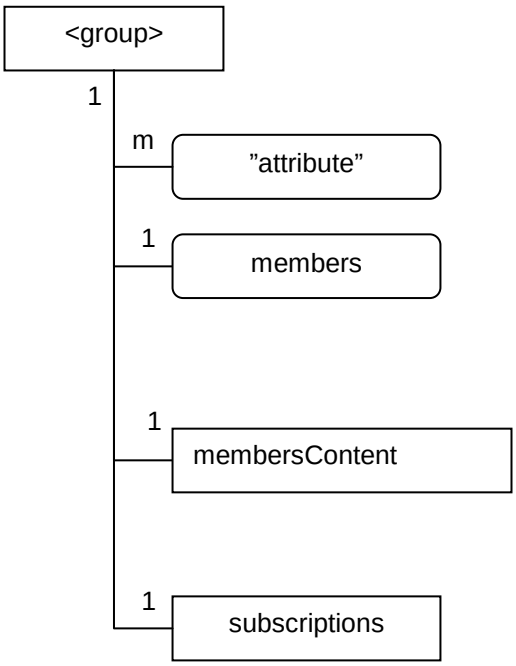


Figure 9.19: Structure of <*group*> resource

The <*group*> resource shall contain the sub resources with the indicated multiplicity in Table 9.33.

Table 9.33

subResource	Multiplicity	Description
membersContent	1	See clause 9.2.3.21.
subscriptions	1	See clause 9.2.3.22.

The <*group*> resource shall contain the attributes that are tagged M (Mandatory) in Table 9.34. The <*group*> resource may also contain attributes that are tagged O (Optional) in Table 9.34.

Table 9.34

AttributeName	Mandatory/Optional	Type	Description
expirationTime	M	RW	See clause 9.2.2 Common attributes.
accessRightID	O	RW	See clause 9.2.2 Common attributes.
searchStrings	M	RW	See clause 9.2.2 Common attributes.
creationTime	M	RO	See clause 9.2.2 Common attributes.
lastModifiedTime	M	RO	See clause 9.2.2 Common attributes.
announceTo	M	RW	See clause 9.2.2 Common attributes.
memberType	M	WO	It is the type of the member resources of the group if all member resources (including the member resources in any sub-groups) are of the same type, otherwise, it is a type of 'mixed'.
currentNrOfMembers	M	RO	Current number of members in a group. It is limited by the *maxNrOfMembers*.
maxNrOfMembers	M	RW	Maximum number of members in the group.
members	M	RW	List of zero or more member URIs referred to in the remaining of the present document as memberID. Each URI (memberID) should refer to a member resource or a (sub-) group resource of the group.
membersContentAccessRightID	O	RW	URI of the accessRight resource defining who is allowed to access the membersContent resource.
consistencyStrategy	O	WO	The attribute determines how to deal with the <group> resource if the memberType validation fails.

9.2.3.20 Resource <groupAnnc>

This resource shall represent an active announcement of a group resource in another SCL. The groupAnnc resource keeps a link to the original resource. That will also be the reference returned during discovery.

It shall be the responsibility of the announcer to keep the accessRight and the searchStrings in sync with the resource that is announced.

The following rules apply:

<sclBase1>/groups/<group> announced on *<sclBase2>* as *<sclBase2>/scls/<scl1>/groups/<groupAnnc>*.

<sclBase1>/applications/<app>/groups/<group> announced on *<sclBase2>* as *<sclBase2>/scls/<scl1>/applications/<appAnnc>/groups/<groupAnnc>*.

Where *<scl1>* is the resource created when *<sclBase1>* registered with *<sclBase1>* and *<appAnnc>* is the announced resource for *<app>*.

The announcing SCL shall suggest the name of the announced resource as the concatenation of the name of the announcing resource with the suffix "Annc", i.e. *<group>Annc*. Or if the *<group>* resource name is "myGroup", the announced resource shall be named "myGroupAnnc". The hosting/announced-to SCL may not accept this name in case of name clashes with locally created group resource. In such a case the announced-to SCL will allocate a different name and return this to the issuer.

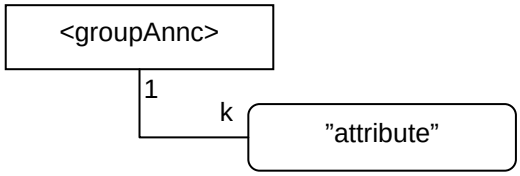


Figure 9.20: Structure of *<groupAnnc>* resource

The *<groupAnnc>* resource shall not contain the any sub resources.

The <*groupAnnc*> resource shall contain the attributes that are tagged M (Mandatory) in Table 9.35. The <*groupAnnc*> resource may also contain attributes that are tagged O (Optional) in Table 9.35.

Table 9.35

AttributeName	Mandatory/Optional	Type	Description
link	M	WO	See clause 9.2.2 Common attributes. The reference to the resource that is announced here.
accessRightID	O	RW	See clause 9.2.2 Common attributes. Same as the *accessRightID* of the announced resource (indicated by the link).
searchStrings	M	RW	See clause 9.2.2 Common attributes. Same as the *searchStrings* of the announced resource (indicated by the link).
expirationTime	M	RW	See clause 9.2.2 Common attributes.

9.2.3.21 Resource membersContent

The *members* attribute of the <*group*> represents a collection of memberIds. Each memberId represents a reference to a resource. The *membersContent* represents the collection of resources identified by these memberIds. Any operation on the *membersContent* shall result in applying that operation on all the members in batch mode. The results of applying that operation to each member in the group shall be aggregated in one result that is returned to the requestor.

The *membersContent* resource does not have a resource representation by itself and consequently it does not have an *accessRightID* attribute. The accessRight resource used for access right validation is indicated by the membersContentAccessRightID attribute in the parent <*group*> resource.

The *membersContent* resource does not represent a real resource in the sense that its representation does not have an e-tag.

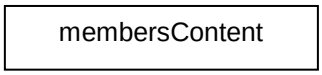


Figure 9.21: Structure of *membersContent* resource

9.2.3.22 Resource subscriptions

For keeping track of subscriptions to a subscribe-able resource, the *subscriptions*-resource is used as a child resource of the subscribe-able parent. The *subscriptions*-resource contains a collection of 0..n <*subscription*> resources that represent individual subscriptions to the subscribable resource (i.e. the parent of the *subscriptions* resource).

The *subscriptions* resource does not have any accessRightID associated with it. The following permissions shall apply.

- READ permission on the parent means CREATE permission to the <*subscription*> resource.
- DELETE permission on the parent means READ and DISCOVERY permission on the *subscriptions* collection resource.

To subscribe to a resource you create a subscription resource in the *subscriptions* collection. You will be notified about the change of the parent resource of the *subscriptions* resource where the <*subscription*> is added.

The structure is as follows.

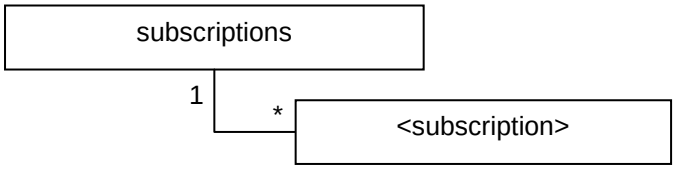


Figure 9.22: Structure of subscribers-resources

The *subscriptions* resource shall contain the sub resources with the indicated multiplicity in Table 9.36.

Table 9.36

subResource	Multiplicity	Description
<subscription>	0..unbounded	See clause 9.2.3.23.

9.2.3.23 Resource <subscription>

A subscription resource shall represent an active (asynchronous) subscription to a resource. I.e. <*resourceURI*>/*subscriptions*/<*subscription*> denotes an active subscription to the resource identified by the resourceURI.

Subscriptions shall only be possible on resources that have a *subscriptions* sub-resource.

The <*subscription*> resource does not have any accessRightID associated with it. The following permissions shall apply.

The subscriber (i.e. the entity that created the subscription) shall have READ, UPDATE, and DELETE permission.

The hosting SCL shall have READ, UPDATE, and DELETE permission.

To subscribe to a resource you create a subscription resource in the corresponding *subscriptions* collection. You will be notified about the change of the parent resource of the *subscriptions* resource where the <*subscription*> is added provided the resource is modified in a way that matches the specified filtercriteria in the subscription. The subscriber shall also be notified (regardless of the filtercriteria) if the parent resource (the subscribed-to resource) is removed.

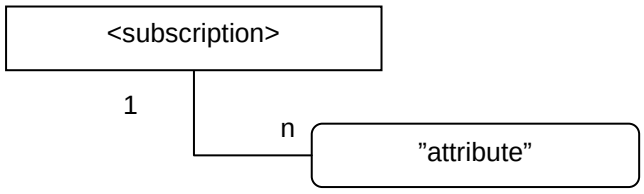


Figure 9.23: Structure of <*subscription*> resource

The <*subscription*> resource shall not contain any sub resources.

The <*subscription*> resource shall contain the attributes that are tagged M (Mandatory) in Table 9.37. The <subscription> resource may also contain attributes that are tagged O (Optional) in Table 9.37.

Table 9.37

AttributeName	Mandatory/Optional	Type	Description
expirationTime	M	RW	See clause 9.2.2 Common attributes.
minimalTimeBetweenNotifications	O	RW	Minimal time between notifications in milliseconds. Only one of minimalTimeBetweenNotifications and delayTolerance may be specified.
delayTolerance	O	RW	Minimal time between event and its notification. Only one of minimalTimeBetweenNotifications and delayTolerance may be specified.
creationTime	M	RO	See clause 9.2.2 Common attributes.
lastModifiedTime	M	RO	See clause 9.2.2 Common attributes.
filterCriteria	O	WO	See clause 9.2.2 Common attributes. This attribute is write-once, which means that if the criteria change, the issuer shall create a new subscription and delete the old one.
subscriptionType	M	RO	Denotes whether it is a long polling or an asynchronous subscription.
contact	M	RW	The URI where the subscriber wants to receive its notifications. This could be a URI of related to a notificationChannel resource, e.g. if the subscriber is not server capable (in the latter case the subscriptionType will be set to synchronous).

9.2.3.24 Resource *m2mPocs*

An *m2mPocs* Collection resource shall represent the collection of <*m2mPoc*> resources that are applicable to the containing <*scl*> resource. The registered SCL can add and remove <*m2mPoc*> resource to and from this collection as its reachability parameters change. See clause 9.2.3.4.

The *m2mPocs* resource is not governed by the normal access right. Only the registering SCL is allowed to create new m2mPoc resources in this collection. Only the registering SCL and the hosting SCL are allowed to retrieve the collection attributes. Also it is not possible to subscribe to this resource.

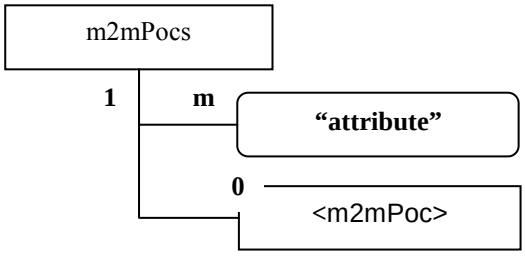


Figure 9.24: Structure of *m2mPocs* resource

The *m2mPocs* resource shall contain the sub resources with the indicated multiplicity in Table 9.38.

Table 9.38

subResource	Multiplicity	Description
m2mPoc	0..unbounded	See clause 9.2.3.25. Zero or more M2M point of contact resources that can be used to reach the SCL for M2M traffic.

The *m2mPocs* resource shall contain the attributes that are tagged M (Mandatory) in Table 9.39. The *m2mPocs* resource may also contain attributes that are tagged O (Optional) in Table 9.39.

Table 9.39

AttributeName	Mandatory/Optional	Type	Description
creationTime	M	RO	See clause 9.2.2 Common attributes.
lastModifiedTime	M	RO	See clause 9.2.2 Common attributes.

9.2.3.25 Resource <*m2mPoc*>

An <*m2mPoc*> Resource represents a possible way to establish a connection in order to reach resources on the <*sclBase*> corresponding to the containing <*scl*>.

The <*m2mPoc*> resource is not governed by the normal access right. Only the registering SCL shall be allowed to update the <*m2mPoc*> resource it created. Only the registering SCL and the hosting SCL shall be allowed to delete or retrieve <*m2mPoc*> resource. Also it is not possible to subscribe to this resource.

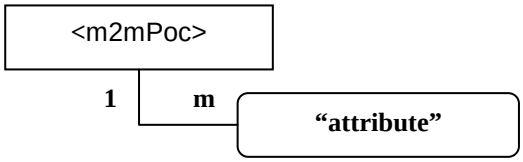


Figure 9.25: Structure of <*m2mPoc*> resource

The <*m2mPoc*> resource shall not contain the any sub resources.

The <*m2mPoc*> resource shall contain the attributes that are tagged M (Mandatory) in Table 9.40. The <*m2mPoc*> resource may also contain attributes that are tagged O (Optional) in Table 9.40.

Table 9.40

AttributeName	Mandatory/Optional	Type	Description
contactInfo	M	RW	Provides information used to obtain routing information for M2M communication. It may be ip-address and port information, it may be a FQDN, or it may be other information that the M2M SP can use to ask the network access provider for a ip-address.
expirationTime	M	RW	Clause 9.2.2 Common attributes This shall represent the expiration time of the m2mPoc resource. If the SCL does not refresh it is the m2mPoc before that time, the m2mPoc is removed. Removal of the m2mPoc may lead to a change of the online status to OFFLINE, see clause 9.2.3.4.
onlineStatus	M	RW	This attribute indicates if the SCL can currently be reached using the contactInfo in this m2mPoc. The remote SCL can set this attribute to OFFLINE to indicate it cannot or does not want to be reached via this m2mPoc, and to ONLINE indicate it can be reached via this m2mPoc. The remote SCL shall never set the value to NOT_REACHABLE. The hosting SCL shall change the value from ONLINE to NOT_REACHABLE, if the hosting SCL detects that the remote SCL cannot be reached via this m2mPoc and it shall change the value from NOT_REACHABLE to ONLINE when the hosting SCL detects the SCL becomes reachable again via this m2mPoc. The mechanism how the hosting SCL determines reachability status of the remote SCL is not defined in the present document. E.g. this may be based on a heartbeat mechanism or on information received from the accessNetwork provider. If the attribute is not provided is shall be set to ONLINE.
creationTime	M	RO	See clause 9.2.2 Common attributes.
lastModifiedTime	M	RO	See clause 9.2.2 Common attributes.

9.2.3.26 Resource *mgmtObjs*

The *mgmtObjs* resource represents a collection of mgmtObj resources and <*mgmtCmd*> resources.

mgmtObjs may be created in multiple places/branches in <*sclBase*> tree of the M2M Server. On <*sclBase*> tree of M2M Server:

- <*sclBase*>/*scls*/<*scl*>/*mgmtObjs* is used to manage service capabilities and for other management functions (network mgmt layer and device management layer) of a single <*scl*> registered to the M2M Server.
- <*sclBase*>/*scls*/<*scl*>/*attachedDevices*/<*attachedDevice*>/*mgmtObjs* is used for management of each D' (or d) device attached to a M2M Gateway (or D device) represented by the <*scl*>.
- <*sclBase*>/*scls*/<*scl*>/*attachedDevices*/*mgmtObjs* is used for M2M Area Network Management of each D' (or d) device attached to a M2M Gateway (or D device) represented by the <*scl*>. The management objects etsiAreaNwkInfo shall be placed here as <*mgmtObj*>. However, etsiAreaNwkDeviceInfo shall be placed under <*sclBase*>/*scls*/<*scl*>/*attachedDevices*/<*attachedDevice*>/*mgmtObjs*.

Creating mgmtObjs in multiple places/braches allows mgmtObjs to explicitly correspond to the management functions (related to application or SCL) with the following benefits:

- Provides flexible control of accessRights for each *mgmtObjs* and its <*mgmtObj*> instances
- More easy to identify/locate a <*mgmtObj*>

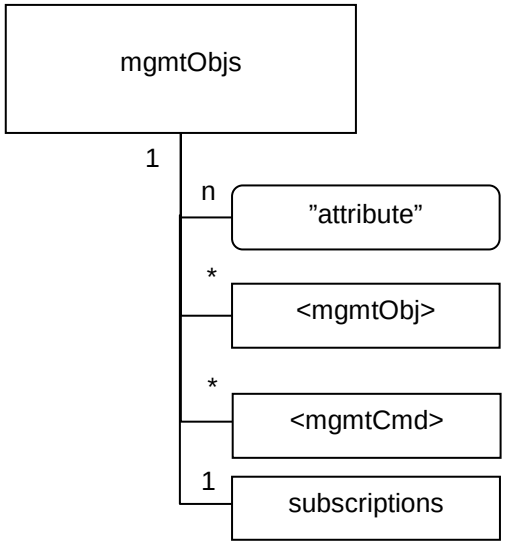


Figure 9.26: Structure of *mgmtObjs* resources

The *mgmtObjs* resource shall contain the sub resources with the indicated multiplicity in Table 9.41.

Table 9.41

subResource	Multiplicity	Description
<mgmtObj>	0..unbounded	See clause 9.2.3.27. Represents a <*mgmtObj*> resource.
<mgmtCmd>	0..unbounded	See clause 9.2.3.29. Represents a <*mgmtCmd*> resource.
subscriptions	1	See clause 9.2.3.22.

The *mgmtObjs* resource shall contain the attributes that are tagged M (Mandatory) in Table 9.42. The *mgmtObjs* resource may also contain attributes that are tagged O (Optional) in Table 9.42.

Table 9.42

AttributeName	Mandatory/Optional	Type	Description
accessRightID	M	RW	See clause 9.2.2 Common attributes.
creationTime	M	RO	See clause 9.2.2 Common attributes.
lastModifiedTime	M	RO	See clause 9.2.2 Common attributes.

9.2.3.27 Resource <*mgmtObj*>

The mgmtObj resource contains management data which represents individual M2M remote entity management functions. It represents a structured Management Object.

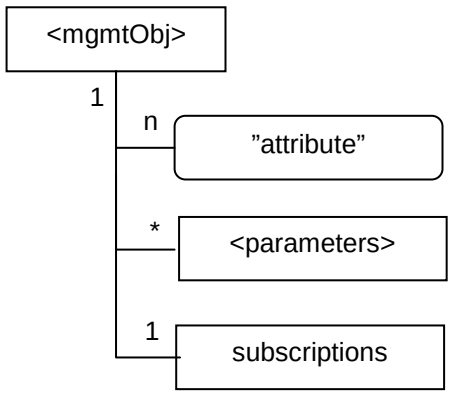


Figure 9.27: Structure of <*mgmtObj*>

The <*mgmtObj*> resource shall contain the sub resources with the indicated multiplicity in Table 9.43.

Table 9.43

subResource	Multiplicity	Description
<parameters>	0..unbounded	Collects multiple parameters or attributes for management purpose. A <*parameters*> can have zero or more other <*parameters*> so as to form hierarchical structure, i.e. the management data for this specific <*mgmtObj*>. Such structure helps to import existing management objects as defined in OMA DM, BBF TR-069, or any other DM technology. For each ETSI specific <*mgmtObj*> resource, the details of <*parameters*> (if any) are specified in annex B. For any other <*mgmtObj*> resource using external data model, <*parameters*> may be partially or completely mapped from the MO data as specified by the corresponding DM technologies (e.g. OMA-DM, BBF).
subscriptions	1	See clause 9.2.3.22.

The <*mgmtObj*> resource shall contain the attributes that are tagged M (Mandatory) in Table 9.44. The <*mgmtObj*> resource may also contain attributes that are tagged O (Optional) in Table 9.44.

Table 9.44

AttributeName	Mandatory/Optional	Type	Description
expirationTime	M	RW	See clause 9.2.2 Common attributes.
accessRightID	M	RW	See clause 9.2.2 Common attributes.
searchStrings	M	RW	See clause 9.2.2 Common attributes.
creationTime	M	RO	See clause 9.2.2 Common attributes.
lastModifiedTime	M	RO	See clause 9.2.2 Common attributes.
moID	M	WO	Contains a URN that uniquely identifies the MO data model used for this <mgmtObj> resource as well as the manage function and version it represents. This attribute shall be provided during the creation of the <mgmtObj> and shall not be modifiable afterwards. It is mandatory for the <mgmtObj>, of which the data model is not specified by ETSI M2M but mapped from underlying existing data model specified in other device management protocols (e.g. OMA-DM, BBF TR-069). For ETSI specific <mgmtObj> resource, this attribute is also mandatory, but the provided value may be ignored, since the MO data model and the management function/version can be determined by its reserved resource name.
originalMO	M	WO	Contains the local path of the original MO instance on the remote entity which is represented by the <mgmtObj> resource in the hosting SCL. This attribute shall be provided during the creation of the <mgmtObj>, so that the hosting SCL can correlate the created <mgmtObj> with the original MO instance on the remote entity for further REM operations. It shall not be modifiable after creation. The format of this attribute shall be a local MO path in the form as specified by existing management protocols (e.g. "./anyPath/Fw1" in OMA-DM, "Device.USBHosts.Host.3." in TR-069).
<moAttribute>	O	RW	Each <moAttribute> is mapped from a leaf node of a MO tree (including ETSI M2M data mode and the existing management data models) based on the mapping rules below the table.
description	O	RW	The text-format description of mgmtObj.

When mapping an MO tree to a corresponding M2M <*mgmtObj*>, the follow rules shall apply:

- The root node of original MO tree (or sub-tree) maps to <*mgmtObj*>:
  - The leaf nodes, if any, of the root node, maps to as the<moAttribute> attribute of the<*mgmtObj*> resource with the same name as the leaf node.
- An interior node of original MO tree maps to a <parameters> with the same name as the interior node.
- A leaf node of an interior node on the original MO tree maps to a<moAttribute> attribute with the same name as the leaf node.

9.2.3.28 Resource <parameters>

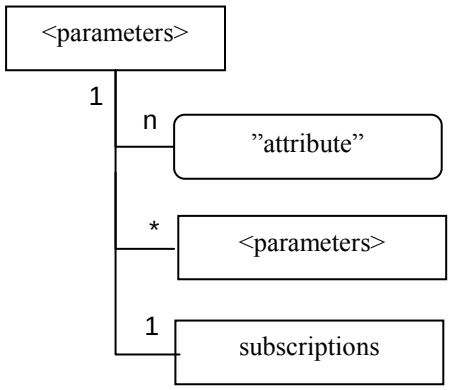


Figure 9.28: Structure of *<parameters>*

The <*parameters*> resource shall contain the sub with the indicated multiplicity in Table 9.45.

Table 9.45

subResource	Multiplicity	Description
<parameters>	0..unbounded	See clause 9.2.3.27.
subscriptions	1	See clause 9.2.3.22.

The <*parameters*> resource shall contain the attributes that are tagged M (Mandatory) in Table 9.46. The <*parameters*> resource may also contain attributes that are tagged O (Optional) in Table 9.46.

Table 9.46

AttributeName	Mandatory/Optional	Type	Description
originalMO	O	WO	See clause 9.2.3.27.
accessRightID	M	RW	See clause 9.2.2 Common attributes.
creationTime	M	RO	See clause 9.2.2 Common attributes.
lastModifiedTime	M	RO	See clause 9.2.2 Common attributes.
<moAttribute>	O	RW	See clause 9.2.3.27.

9.2.3.29 Resource *<mgmtCmd>*

<mgmtCmd> resource models non-RESTful management commands or RPC in BBF TR-069. It enables that an M2M Network Application can request to execute a non-RESTful RPC on a remote entity or even cancel an initiated yet unfinished cancellable management command over mIa reference point in a RESTful manner.

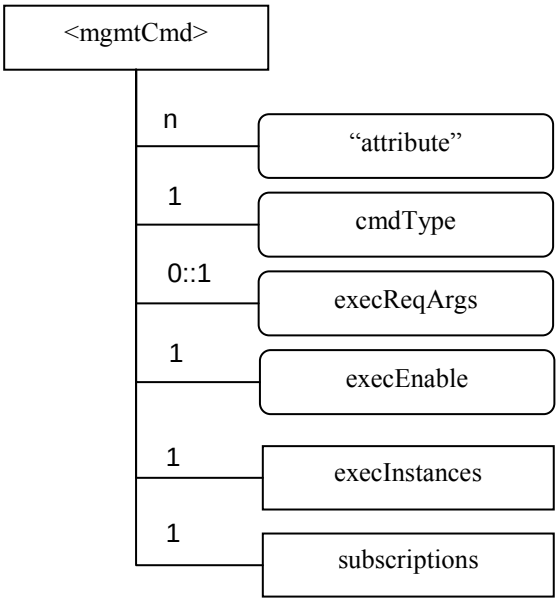


Figure 9.29: Structure of resource <*mgmtCmd*>

- <mgmtCmd> shall be placed as a sub-resource of mgmtObjs. Multiple <mgmtCmd> can appear as a sub-resource of mgmtObjs. Each <mgmtCmd> corresponds to a specific type of management commands, as defined by its attribute cmdType. There might be multiple requests to the same <mgmtCmd>. As a result, <mgmtCmd> has a sub-resource execInstances to contain all execution instances of the <mgmtCmd>. <mgmtCmd> shall be triggered using UPDATE method to its attribute "execEnable".

- Figure 9.29 shows the structure of <mgmtCmd> resource. The commands shall contain the following sub-resources.

Table 9.47

subResource	Mandatory/Optional	Multiplicity	Description
execInstances	M	1	A collection resource as the placeholder to contain all execution instances of the same <*mgmtCmd*>.
subscriptions	M	1	See clause 9.2.3.22.

- and the following attributes.

Table 9.48

AttributeName	Mandatory/Optional	Type	Description
expirationTime	M	RW	See clause 9.2.2 Common attributes.
accessRightID	M	RW	See clause 9.2.2 Common attributes.
searchStrings	M	RW	See clause 9.2.2 Common attributes.
creationTime	M	RO	See clause 9.2.2 Common attributes.
lastModifiedTime	M	RO	See clause 9.2.2 Common attributes.
description	O	RW	The text-format description of this resource
cmdType	M	WO	The type to identify the command (e.g. urn:bbf:tr069:download).
execReqArgs	O	RW	Structured attribute (e.g. abstract type) to contain any command-specific arguments of the request.
execEnable	M	RO	The attribute can be blank without any value or it can contain a URI that can be used to trigger execution of <*mgmtCmd*> using UPDATE method.

9.2.3.30 Resource execInstances

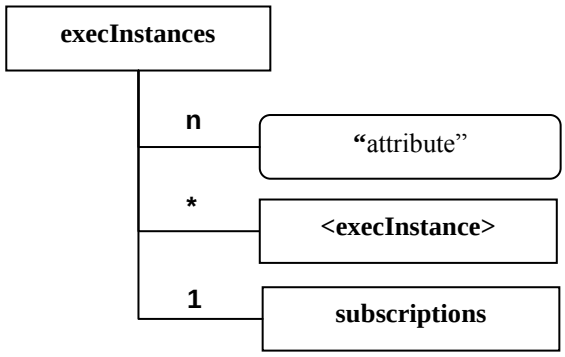


Figure 9.30: Structure of resource execInstances

- execInstances contains all execution instances of the same management command <mgmtCmd>. Each <execInstance> is a sub-resource of execInstances. When <mgmtCmd> is successfully triggered by an M2M network application using UPDATE method, a corresponding <execInstance> shall be generated.
- Figure 9.30 shows the structure of resource execInstances. The execInstance resource shall contain the following sub-resources.

Table 9.49

subResource	Mandatory/Optional	Multiplicity	Description
<execInstance>	M	0::unbounded	The execution instance of *<mgmtCmd>.*
subscriptions	M	1	See clause 9.2.3.22.

- and the following attributes.

Table 9.50

AttributeName	Mandatory/Optional	Type	Description
creationTime	M	RO	See clause 9.2.2 Common attributes.
lastModifiedTime	M	RO	See clause 9.2.2 Common attributes.

9.2.3.31 Resource <execInstance>

- This resource represents an ongoing execution request instance, triggered by an M2M network application using UPDATE method to the attribute "execute" of <*mgmtCmd*>.

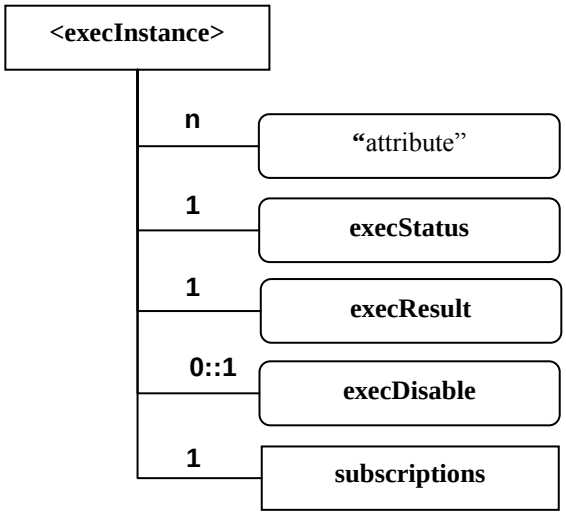


Figure 9.31: Structure of resource <execInstance>

- Figure 9.31 shows the structure of resource <execInstance>. The <execInstance> resource shall contain the following sub-resources.

Table 9.51

subResource	Mandatory/Optional	Multiplicity	Description
subscriptions	M	1	See clause 9.2.3.22.

- and the following attributes.

Table 9.52

AttributeName	Mandatory/Optional	Type	Description
expirationTime	M	RO	See clause 9.2.2 Common attributes.
accessRightID	M	RW	See clause 9.2.2 Common attributes.
creationTime	M	RO	See clause 9.2.2 Common attributes.
lastModifiedTime	M	RO	See clause 9.2.2 Common attributes.
execStatus	M	RO	The status of <*execInstance*>. It can be initiated, started, finished, cancelled, or deleted.
execResult	M	RO	The execution result of <*execInstance*>.
execDisable	O	RW	The attribute is used to cancel <*execInstance*> using UPDATE method, Some BBF TR-069 RPCs such as Reboot cannot be cancelled and accordingly this attribute is optional.

9.2.3.32 Resource attachedDevices

This type of resources is used to collect the management information of all M2M D' (or d) devices that are attached to a M2M Gateway (or D device). It shall reside under a G (or D) <*scl*> resource created in the remote NSCL.

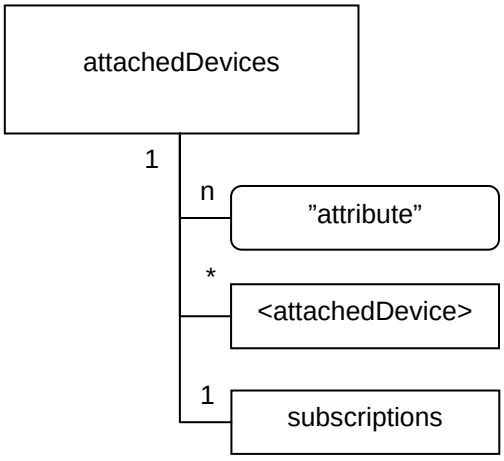


Figure 9.32: Structure of *attachedDevices* resource

The *attachedDevices* resource shall contain the sub resources that are tagged M (Mandatory) with the indicated multiplicity in Table 9.53. The *attachedDevices* resource may also contain sub resources that are tagged O (Optional) with the indicated multiplicity in Table 9.53.

Table 9.53

subResource	Mandatory/Optional	Multiplicity	Description
<attachedDevice>	O	0..unbounded	See clause 9.2.3.33. Represents a attachedDevice resource.
subscriptions	M	1	See clause 9.2.3.22.

The *mgmtObjs* resource shall contain the attributes that are tagged M (Mandatory) in Table 9.54. The *attachedDevices* resource may also contain attributes that are tagged O (Optional) in Table 9.54.

Table 9.54

AttributeName	Mandatory/Optional	Type	Description
accessRightID	M	RW	See clause 9.2.2 Common attributes.
creationTime	M	RO	See clause 9.2.2 Common attributes.
lastModifiedTime	M	RO	See clause 9.2.2 Common attributes.

9.2.3.33 Resource <attachedDevice>

This type of resources is used to represent an M2M D' (or d) device that is attached to a M2M Gateway (or D device). It is a resource in the NSCL that shall reside under the *AttachedDevices* resource of the corresponding M2M Gateway (or D device).

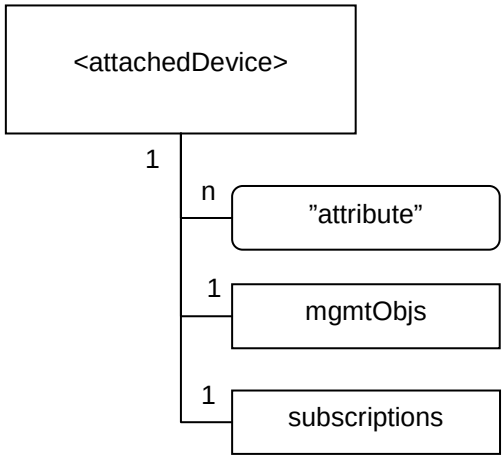


Figure 9.33: Structure of <*attachedDevice*> resource

The *mgmtObjs* resource shall contain the sub resources that are tagged M (Mandatory) with the indicated multiplicity in Table 9.55. The <*attachedDevice*> resource may also contain sub resources that are tagged O (Optional) with the indicated multiplicity in Table 9.55.

Table 9.55

subResource	Mandatory/Optional	Multiplicity	Description
mgmtObjs	M	1	See clause 9.2.3.26. Collects the <*mgmtObj*> resources of an attached device represented by the parent <*attachedDevice*> resource.
subscriptions	M	1	See clause 9.2.3.22.

The *attachedDevice* resource shall contain the attributes that are tagged M (Mandatory) in Table 9.56. The *attachedDevice* resource may also contain attributes that are tagged O (Optional) in Table 9.56.

Table 9.56

AttributeName	Mandatory/Optional	Type	Description
accessRightID	M	RW	See clause 9.2.2 Common attributes.
creationTime	M	RO	See clause 9.2.2 Common attributes.
lastModifiedTime	M	RO	See clause 9.2.2 Common attributes.

9.2.3.34 Resource notificationChannels

The *notificationChannels* resource represents a collection of notification channels. This resource can be a sub-resource of a registered application or a registered SCL. I.e. <*sclBase*>*/applications/*<*application*>*/notificationChannels* or <*sclBase*>*/scls/*<*scl*>*/notificationChannels*.

The *notificationChannels* resource is a resource that is only to be used by the corresponding application or SCL. Therefore, the *notificationChannels* resource does not have any accessRightID associated with it. Only the default accessRights apply (i.e. only the entities in the path, or ancestor entities, have all access rights).

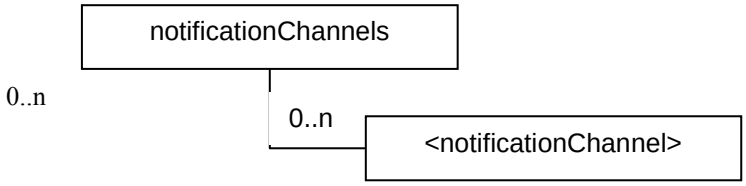


Figure 9.34: Structure of *notificationChannels* resource

The *notificationChannels* resource shall contain the sub resources with the indicated multiplicity in Table 9.57.

Table 9.57

SubResource	Multiplicity	Description
<notificationChannel>	0..unbounded	See clause 9.2.3.35.

The *notificationChannels* resource shall contain the attributes that are tagged M (Mandatory) in Table 9.58. The *notificationChannels* resource may also contain attributes that are tagged O (Optional) in Table 9.58.

Table 9.58

AttributeName	Mandatory/Optional	Type	Description
creationTime	M	RO	See clause 9.2.2 Common attributes.
lastModifedTime	M	RO	See clause 9.2.2 Common attributes.

9.2.3.35 Resource <notificationChannel>

The <*notificationChannel*> resource introduces a method for a non-server capable client to retrieve asynchronous notifications for which the client has subscribed. The notification channel is prepared to handle several mechanisms on how to receive these asynchronous notifications. However, currently only one mechanism is fully specified, which is the so-called "long polling" mechanism. This method is based on the server not responding to requests until a notification needs to be sent (or until a timeout occurs).

The notification channel resource has a contactURI which is used in subscriptions. In the case of long polling channeltype, the channelData contains another URI that is to be used for long polling. The notification channel links these two URIs together, such that notification that are received on the provided contactURI are returned as responses to long polling requests happening on the corresponding long polling URI.

See clause 9.3.2.2.26.6 for more details on long polling.

Currently, only the long polling channelType is defined and the channelData associated with this type.

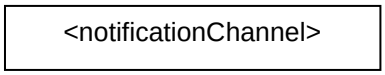


Figure 9.35: Structure of <*notificationChannel*> resource

The <*subscription*> resource shall not contain the any sub resources.

The <*notificationChannel*> resource shall contain the attributes that are tagged M (Mandatory) in Table 9.59. The <*notificationChannel*> resource may also contain attributes that are tagged O (Optional) in Table 9.59.

Table 9.59

AttributeName	Mandatory/Optional	Type	Description
channelType	M	RW	The type of the notificationChannel. Currently only longPolling is supported.
contactURI	M	RO	The URI that is used in subscriptions.
channelData	M	RO	The data associated with the channel. The type of data may differ depending on the channelType. For the longPolling channelType, the channelData includes a URI on which the client can do the long polling request in order to get the notifications that were sent to the contactURI.
creationTime	M	RO	See clause 9.2.2 Common attributes.
lastModifiedTime	M	RO	See clause 9.2.2 Common attributes.

9.2.3.36 Resource discovery

The discovery resource shall be used to perform a discovery of resources matching specified filterCriteria under the <*sclBase*> and provide this result back to the issuer.

It does not represent a real resource in the sense that its representation does not have an e-tag.

The discovery resource does not have an accessRightID attribute. Every authenticated entity shall have READ permissions. However, the result that is returned to the issuer shall only contain URIs of resources for which the issuer has any permission (which could be limited to only DISCOVER permission).

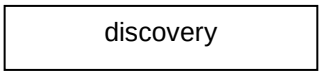


Figure 9.36: Structure of *discovery* resource

9.3 Interface Procedures

9.3.1 General concept and procedures

9.3.1.1 General responses

The accepted responses for the dIa, mIa and mId reference points are:

- Success: indicates to the issuer that the request is executed successfully by the SCL.
- Failure: indicates to the issuer that the request has not been executed successfully by the SCL.

Both success and failure indication may convey further information.

Detailed success and failure codes are provided in TS 102 921 [1].

In addition to the above mentioned responses, there is another response possible that indicates an Acknowledgement of the request. This indicates to the issuer that the request has been received by the SCL, but not yet executed. The success or failure of the request is later conveyed (either synchronously or asynchronously). See clause 9.3.1.4 for details. This response is not used in the subsequent clause 9.3.2.

9.3.1.2 General mechanisms

This clause describes the mechanisms that are used throughout the document for:

- Accessing resources that may be located in different SCLs.
- Synchronous and asynchronous response mechanism.
- Aggregation of requests to access remotely hosted resources by store-and-forward handling.

9.3.1.3 Accessing resources in SCLs

For all procedure described in the following clauses, the addressed resource can be stored in different SCLs. Table 9.60 describes the possible cases.

Table 9.60

Name	Description	Figure
no hop	• The issuer is either an application or a SCL. • The issuer accesses a resource, the local SCL and hosting SCL are the same entity. • The SCL shall check the access rights for the resource. • The SCL shall respond to the issuer accordingly, either with a successful response or with an error.	Figure 9.37 for accessing a resource.
1 hop	• Issuer may only be an Application. • The issuer accesses a resource. • The Local and the hosting SCL are different. • The Local SCL shall forward the request to the Hosting SCL, after an optional checking of access rights and syntax. • Then the hosting SCL after checking the access right shall respond with a successful or error response.	Figure 9.38 for accessing a resource.
2-hop	• Issuer may be an application or a SCL. • The issuer accesses a resource. • The local SCL, intermediate SCL and the hosting SCL are different. • The local SCL shall forward the request to an intermediate SCL (e.g. NSCL) that the local SCL registered with, if it cannot communicate with the hosting SCL directly. An optional checking of access rights and syntax can be performed prior to forwarding. • The intermediate SCL shall forward the request to the hosting SCL. An optional checking of access rights and syntax can be performed prior to forwarding. • The hosting SCL shall check the access rights and respond with an appropriate response.	Figure 9.39 for accessing a resource.

NOTE 1: In case of group communication, a multi-hop case may also be supported as described in clause 9.3.2.17.

NOTE 2: Security association is established hop-by-hop individually between NSCL and D/GSCL.

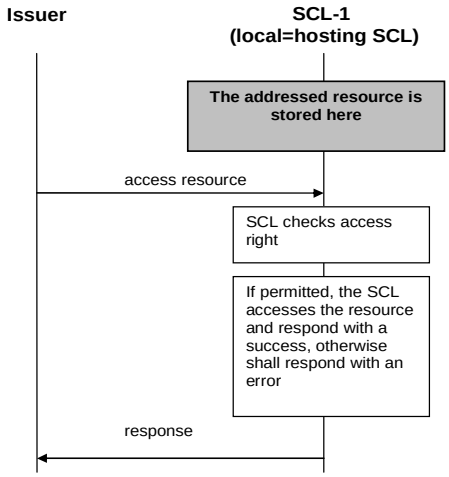


Figure 9.37: Issuer accesses a resource in the Local SCL; no hop

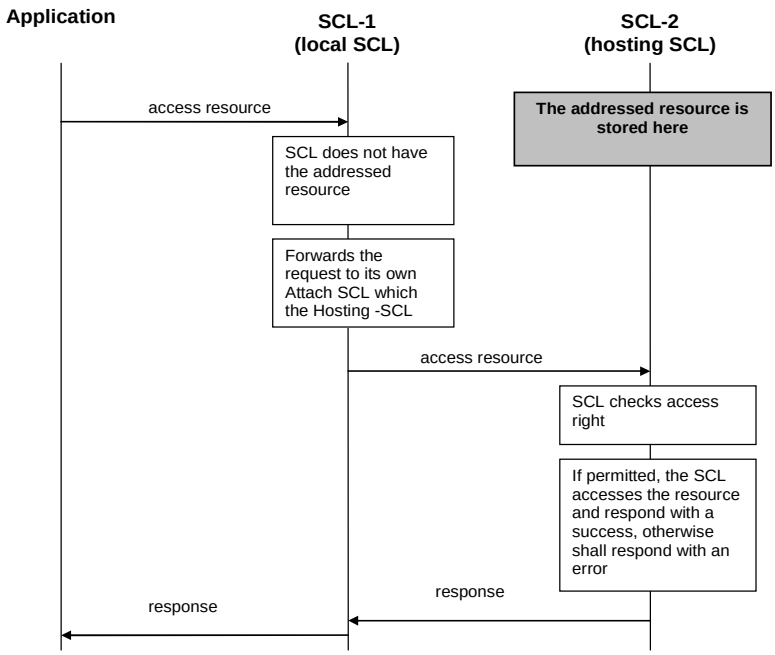


Figure 9.38: Application accesses resource in the hosting SCL; one hop

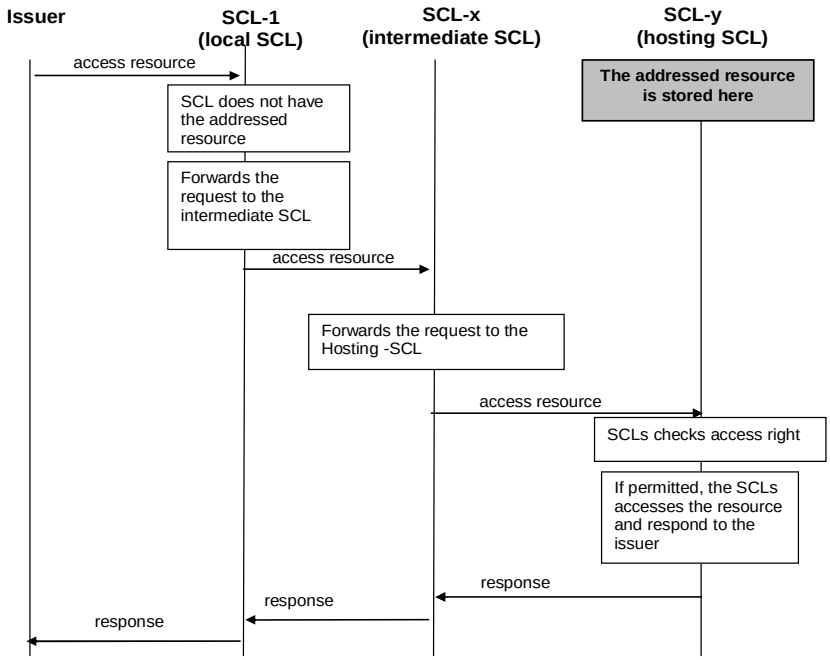


Figure 9.39: Issuer accesses a resource in the hosting SCL; 2-hop

The generic procedures on both the Local SCL and the Announced-to SCL are applicable to all the cases described in the following clauses.

9.3.1.4 Client-2-server and server-2-server communication

There are several mechanisms for reporting responses to a request issuer. These mechanisms are listed in Table 9.61.

Table 9.61

<table>
<tr><th>Mechanism</th><th colspan="2">Description</th><th>Issuer type (client/server)</th><th>Receiver type (client/server)</th><th>Figure number</th></tr>
<tr><td>Client-2-Server (Synchronous)</td><td colspan="2">the issuer sends a request and it shall receive the requested information directly in the response to the request</td><td>• The issuer shall be a client.
• The issuer may also be server capable.
This is the most use mechanism, where the result of the request is sent in the same connection.
• The issuer is blocked until the result is received.</td><td>The receiver shall be a server</td><td>Figures 9.40 and 9.41

Figures 9.43 and 9.44</td></tr>
<tr><td>Client-2 -Server (Semi-asynchronous)</td><td rowspan="2">the issuer sends a request and it shall only get a confirmation that the request has been received and will be processed.
The issuer shall be informed of the result of the request in one of the following ways:</td><td>by actively fetching the information (short polling mechanism).</td><td>• The issuer shall be a client.
• The issuer may also be server capable.
• This mechanism may not be very efficient due to the possible high signalling.
However it may be useful in case that the receiver requires some computation and it might not be able to process the request in a short time.</td><td rowspan="2">The receiver shall be a server</td><td>Figures 9.40 and 9.42</td></tr>
<tr><td></td><td>by actively polling for the information (long polling mechanism).
In this case the request shall contain an indication that the issuer is performing a long polling.</td><td>• The issuer shall be a client.
• The issuer may also be server capable.
• This mechanism is mainly used for subscribing to changes of a resource when the issuer is not able to receive notification and therefore is mainly when the issuer is only a client.</td><td></td></tr>
<tr><td>Server-2-Server (Asynchronous)</td><td colspan="2">the issuer sends a request and it shall only get a confirmation that the request has been received and it will be processed.
The result will be sent to the issuer with a new request coming from the receiver of the request.</td><td>• The issuer shall be both client and server.</td><td>The receiver shall be both client and server</td><td>Figures 9.43 and 9.45</td></tr>
</table>

The following figures describe the mechanisms.

Client-2-Server

The issuer sends a request containing the needed parameters and a special parameter: "correlation_data" is used to uniquely identify the request within an interaction. (For an interaction is meant to identify all messages that are exchanged from the moment that an original request is sent and the final response is received), see Figure 9.40.

Figure 9.40: Client-2-server mode Request

Then the receiver shall either sends an immediate response containing the results and the interaction is closed, see Figure 9.41.

Figure 9.41: Client-2-server mode immediate Response

Or the receiver shall send an acknowledge indication of the request.

In this case the issuer, in order to get a response, shall send the original request to the receiver, containing the exact same data including the "correlation_data" (to let the receiver understand that this is an "old" request).

If the results are not available, the receiver shall send an acknowledgement again, otherwise it shall send a response and the interaction is closed. See Figure 9.42.

The interaction is also closed when the results are available on the receiver side, but the issuer does not send a request to get them in a predefined interval of time.

A request containing the same parameters and "correlation_data", for a previously closed interaction, shall be treated as a "new" request.

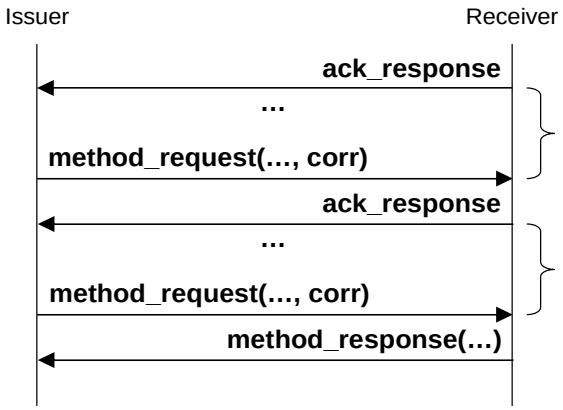


Figure 9.42: Client-2-server mode delayed Response

NOTE: The possible presence of a Load Balancing function between the client and the server is to be taken in account for polling mechanics.

The following considerations shall be taken in account in the mechanisms described above.

The receiver may decide if the answer to a request shall be sent synchronously, with an immediate result (see Figure 9.41) or if it shall only send an acknowledge indication (see Figure 9.42). Therefore the issuer shall be able to understand both types of responses.

If the receiver (server) sends only an acknowledge indication, it shall also provide to the issuer the minimum time to wait before re-sending the request. In this scenario, the client may also go off-line between request acknowledge and a new request.

It is up to the server to decide for how long the results shall be available before closing the interaction.

Server-2-server

The issuer shall send a request containing the needed parameters and two special parameters: "correlation_data" and "contact_server", see Figure 9.43. The request is very similar to the previous one depicted in Figure 9.40, however in this case the issuer provides the extra parameter for indicating the address (still in the issuer domain) where the results shall be sent.

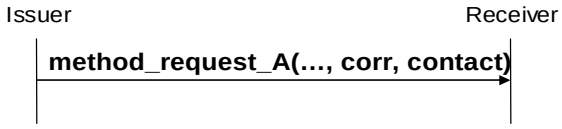


Figure 9.43: Server-2-server mode Request

Then the receiver shall either send an immediate response containing the results and the interaction is closed, see Figure 9.44. This response shall be exactly the same as the one indicated in Figure 9.41 for the same request.

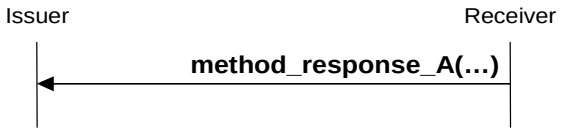


Figure 9.44: Server-2-server mode immediate Response

Or the receiver shall send an acknowledge indication to the request.

In this case the receiver shall send the results of the request to the issuer in a new method request. The new request is sent to the "contact_server" address received in the original request. The request shall contain the "correlation_data" (to let the receiver understand that this is the result to a previously sent request). The interaction is closed after the results are received by the original issuer.

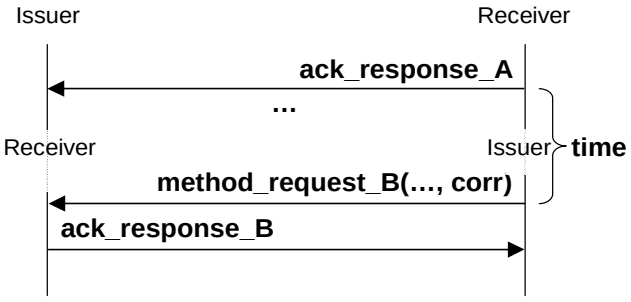


Figure 9.45: Server-2-server mode delayed Response

The issuer may send again the original request to the receiver, containing the same "correlation_data" and "contact_server" (to let the receiver understand this is re-transmission of an "old" request). If the results are not available because the previous request is still in progress, the server shall send an acknowledge indication, see Figure 9.46.

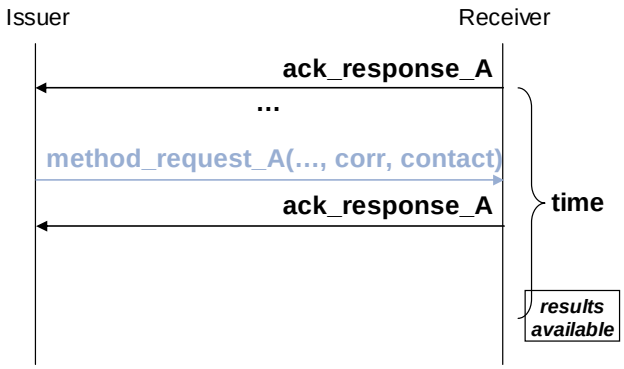


Figure 9.46: Server-2-server mode request re-issued

If the results are available and have not yet been successfully sent, the receiver shall send a response, see Figure 9.47. Note that between the acknowledge indication and the new request, the issuer could have been also off-line.

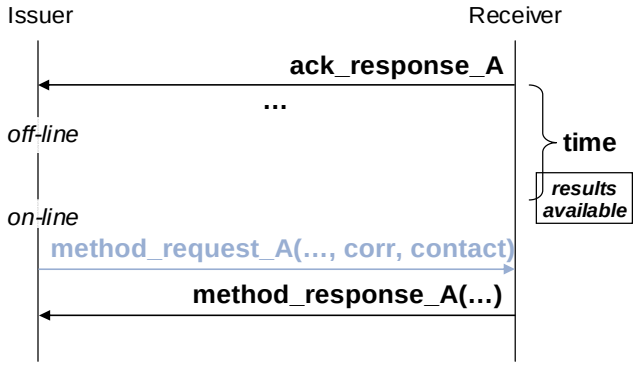


Figure 9.47: Server-2-server mode request re-issued when on-line

If the results have been discarded because the receiver has already sent them or it has not been able to successfully send them for a predefined interval of time, it shall treat the request as a "new" one, see Figure 9.48.

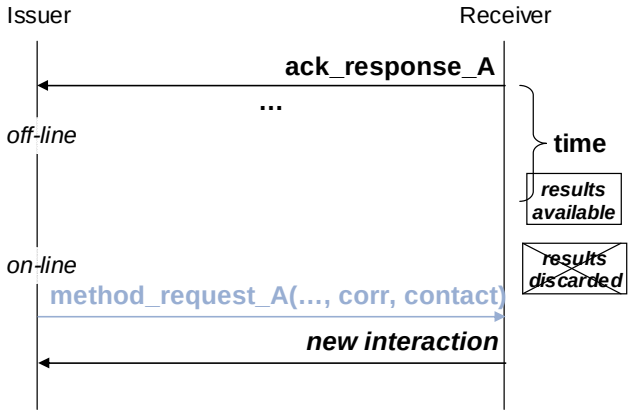


Figure 9.48: Server-2-server mode request re-issued when online

If the results have been already sent to the issuer the interaction is closed, therefore any request containing the same "correlation_data" and "contact_server" is considered as new request, see Figure 9.49.

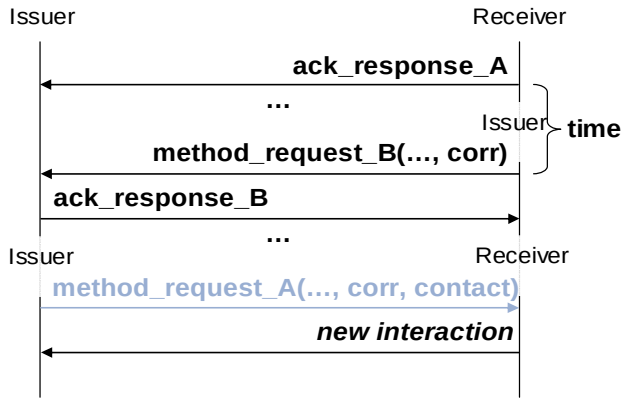


Figure 9.49: Server-2-server mode request re-issued after results are received

The following considerations shall be taken in account in the mechanisms described for the server-2 server communication.

The receiver may decide if the answer to a request shall be sent synchronously, with an immediate result (see Figure 9.44) or if it shall only send an acknowledge indication (see Figure 9.45). Therefore the issuer shall be able to understand both types of responses.

The "contact_server" may also go off-line, in this case the receiver shall wait for it to be on-line in order to send the request with the results. However it is up to the receiver to decide how long to wait for a "contact_server" to be on-line before closing the interaction and discard the results.

9.3.1.5 Aggregation of requests to access remotely hosted resources by store-and-forward handling

9.3.1.5.1 General principle of store-and-forward handling for accessing remotely hosted resources

The following stage 2 description of store-and-forward handling applies to requests to access remotely hosted resources irrespective of whether the source of the request is on the network or M2M Device/Gateway side, i.e. it applies to requests issued by a NA or NSCL to access resources hosted on a D/GSCL as well as to requests issued by a D/GA or D/GSCL to access resources on a NSCL.

For each request to access remotely hosted resources - i.e. requests according to the one or more hop case explained in clause 9.3.1.3 - the request issuer may indicate a Tolerable Request Processing Delay Time (TRPDT) and/or a request category (RCAT).

The indication of TRPDT allows the local SCL that receives the request to delay forwarding of the request to the hosting SCL up to the time indicated by TRPDT - either as an absolute time or as a delay period - at the benefit of possibly aggregating more requests to the same hosting SCL. The amount of time that the local SCL actually waits with forwarding of the request shall be governed by policies but is limited by the TRPDTs of the pending requests. These policies are a result of a consolidation process (policy administration functions) on the NSCL. Both the M2M Service Provider and Access Network provider can supply policies in the NSCL defined for an application or for a SCL. Access network provider policy is mandatory. These policies are stored in the NSCL; the mechanism for provisioning these policies in the NSCL is out of scope of the present document. These policies may be specified on the application level, on the D/GSCL level or on the NSCL level. The NSCL shall consolidate these policies as follows:

- if present, use the application level policy; otherwise
- if present, use the D/GSCL level policy; otherwise
- use the NSCL level policy, which is the default policy.

The NSCL has the responsibility to consolidate policies from the M2M Service Provider and the Access Network providers. The NSCL shall then provision the consolidated policy into a D/GSCL by using either xREM procedures via Management Objects as defined in clauses 9.3.2.23 and B.1 under SCL Management via the resource "etsiSclMo" defined in clause B.2.1 or by mechanisms that are out of scope of the present document (e.g. pre-provisioning). The M2M Service Provider shall be authorized to provision the policies that govern the actual waiting time for aggregating requests.

The indication of RCAT allows the local SCL to block forwarding of requests to access remotely hosted resources if the SCL cannot establish connectivity using an appropriate access network for the given RCAT value, the given issuer, and (in case of NSCL) the given destination. The definition of when it is appropriate to use a specific access network and/or to attempt to establish connectivity for a given RCAT value (i.e. the schedule) shall be governed by policies. Such policies shall be provisioned into a D/GSCL by using either xREM procedures via Management Objects as defined in clauses 9.3.2.23 and B.1 under SCL Management via the resource "etsiSclMo" defined in clause B.2.1or mechanisms that are out of scope of the present document. The policies provisioned to the D/GSCL are a consolidated version of the policies provisioned on the NSCL as described above. In this process the NSCL may allocate a D/GSCL-specific schedule within the allowed schedule provisioned to the NSCL and provision this consolidated and down-selected schedule to each D/GSCL for congestion control.

Selection of a specific access network among those that are determined to be appropriate at a given time, for a given RCAT, a given issuer, and a given destination (in case of NSCL processing the request), may depend on policies that consider cost, power consumption, available bandwidth, etc. Such policies shall be provisioned by the NSCL into a D/GSCL after consolidation by using either xREM procedures via Management Objects as defined in clauses 9.3.2.23 and B.1 under SCL Management via the resource "etsiSclMo" defined in clause B.2.1 or by mechanisms that are out of scope of the present document.

The granularity of possible TRPDT values and the number of categories for RCAT values are specified in [1].

The local SCL that receives a request to access remotely hosted resources shall do local checking of the validity of the request to decide about the need to forward request or potentially return an early response. In case forwarding is needed, the local SCL shall then use store-and-forward (SAF) handling to possibly aggregate multiple requests and/or forward requests to the target SCLs when appropriate connectivity can be used. The behaviour of a local SCL following SAF handling is described in the following clauses.

9.3.1.5.2 Request issuer indicates no TRPDT and no RCAT

In case no TRPDT and no RCAT is indicated by the issuer of the request, default values for TRPDT and RCAT will be used and the behaviour of the local SCL shall be as described in clause 9.3.1.5.5 for the case that the issuer indicated a combination of TRPDT and RCAT. Policies on default values for TRPDT and RCAT shall be provisioned into a D/GSCL by using either xREM procedures via Management Objects as defined in clauses 9.3.2.23 and B.1 under SCL Management via the resource "etsiSclMo" defined in clause B.2.1 or by mechanisms that are out of scope of the present document (e.g. pre-provisioning). The M2M Service Provider shall be authorized to provision policies for default values for TRPDT and RCAT.

9.3.1.5.3 Request issuer indicates TRPDT only

In case only TRPDT is indicated by the issuer of the request, a default value for RCAT will be used and the behaviour of the local SCL shall be as described in clause 9.3.1.5.5 for the case that the issuer indicated a combination of TRPDT and RCAT.

9.3.1.5.4 Request issuer indicates RCAT only

In case only RCAT is indicated by the issuer of the request, a default value for TRPDT will be used and the behaviour of the local SCL shall be as described in clause 9.3.1.5.5 for the case that the issuer indicated a combination of TRPDT and RCAT.

9.3.1.5.5 Request issuer indicates a combination of TRPDT and RCAT

The behaviour of the local SCL that receives the request shall be as follows:

1) If connectivity is already established at the time a request is issued to the local SCL and if one of the used access networks is determined to be appropriate for the given RCAT value and the given issuer and destination (in case NSCL processing the request), the local SCL shall select one of the appropriate access networks with already established connectivity and forward the request to the target SCL and deliver the response accordingly.

2) If currently no connectivity is established, the local SCL shall check whether the request shall be forwarded immediately (TRPDT indicates now) or whether it may be forwarded at a later time (TRPDT indicates a later time).

3) If the request shall be forwarded immediately, the local SCL will continue with forwarding the request without waiting for additional events and continues with step 6.

4) The local SCL shall acknowledge the reception of the request as described in clause 9.3.1.4 for semi-asynchronous (client-2-server) or asynchronous (server-to-server) communication mechanisms.

5) The local SCL may wait with forwarding the request to the target SCL up to the time indicated by TRPDT.

6) The local SCL shall check if establishment of connectivity via an access network that is appropriate for the given RCAT value of the request and the given issuer and destination (in case NSCL processing the request) is currently possible.

7) If establishment of connectivity via an appropriate access network is currently not possible; and

   a) the delay limit given by TRPDT has not expired, the local SCL shall continue with step 5;

   b) the delay limit given by TRPDT has expired, the local SCL shall respond with a failure response to the issuer:

      i) according to the semi-asynchronous (client-2-server) or asynchronous (server-to-server) communication mechanisms described in clause 9.3.1.4 if the request was acknowledged in step 4;

      ii) according to the default response mechanism in case step 4 was skipped.

8) If establishment of connectivity via an appropriate access network is possible, the SCL shall select one of the appropriate access networks, establish connectivity, initiate the connection(s) to the remote SCL(s), forward the aggregated request(s) in the order received and respond to the issuer(s) with the response(s) from the remote SCL(s):

   a) using the semi-asynchronous (client-2-server) or asynchronous (server-to-server) communication mechanisms described in clause 9.3.1.4 in case step 4 was executed for this request;

   b) using the default response mechanism in case step 4 was skipped.

9.3.1.5.6 Policies governing SAF handling

9.3.1.5.6.1 Policies that the access network provider is authorized to provision in the NSCL

The types of policies that a local SCL shall support when determining whether to attempt to use the access network of a specific access network provider in step 6 of clause 9.3.1.5.5 are as follows:

1) Schedule of RCAT values versus time
   The access network provider shall be authorized to provision information to the NSCL about when it is appropriate to forward requests of a given RCAT value. The resources that are used to provision such scheduling policies in case xREM procedures are used, is defined in clause B.2 as part of the resource "etsiSclMo" defined in clause B.2.1.

2) Blocking of access attempts after failure to establish connectivity
The access network provider shall be authorized to provision information to the local SCL about a period of time over which attempts to establish connectivity over its access network are not appropriate after the previous attempt to establish connectivity over the corresponding access network has failed. The period of time to block attempts to establish connectivity can be a function of the number of consecutive previous attempts to establish connectivity over this access network. The resources that are used to provision such blocking policies in case xREM procedures are used, are defined in clause B.2 as part of the resource "etsiSclMo" defined in clause B.2.1.

Both types of policies can be provisioned to the NSCL either as application-specific policies(identified by an APP-ID and a SCL-ID) or as D/GSCL-specific (identified by a SCL-ID) or as default policies.

The support of policies is mandatory. The policies shall be applied, according to the following:

On the NSCL:

- if present the application-specific policies for the request issuer and the request destination SCL shall be used; otherwise
- if present, the D/GSCL-specific policies for request destination SCL shall be used; otherwise
- if present, default policies shall be used; otherwise
- the request is rejected.

On the D/GSCL the consolidated policies provisioned by the NSCL shall be used as follows:

- if present the application-specific policies for the request issuer shall be used; otherwise
- if present, default policies shall be used; otherwise
- the request is rejected.

9.3.1.5.6.2 Policies that the M2M Service Provider is authorized to provision

The types of policies that a local SCL shall support when determining whether to forward a request to target SCL(s) are as follows:

1) Wait time as function of number of pending requests
The M2M Service Provider shall be authorized to provision information to the local SCL about how many pending requests of a given range of RCAT values are sufficient to continue after step 4 in clause 9.3.1.5.5. The ranges of RCAT values for different policies shall not overlap The resources that are used to provision policies for wait time as function of number of pending requests in case xREM procedures are used, are defined in clause B.2 as part of the resource "etsiSclMo" defined in clause B.2.1.

2) Wait time as function of amount of pending request data
The M2M Service Provider shall be authorized to provision information to the local SCL about a threshold of consumed storage in the local SCL that is needed to buffer data for pending requests of a given range of RCAT values that should trigger continuation after step 4 in clause 9.3.1.5.5. The ranges of RCAT values for different policies shall not overlap The resources that are used to provision policies for wait time as function of amount of pending request data in case xREM procedures are used, is defined in clause B.2 as part of the resource "etsiSclMo" defined in clause B.2.1.

3) Selection among appropriate access networks
The M2M Service Provider shall be authorized to provision information to the local SCL about how to select an access network for making an attempt to establish connectivity among all appropriate access networks in step 1 and 8 in clause 9.3.1.5.5. Each of these policies shall define a ranking of preferable access networks for a given range of RCAT values. The ranges of RCAT values for different policies shall not overlap. The resources that are used to provision policies for selection among appropriate access networks in case xREM procedures are used, are defined in clause B.2 as part of the resource "etsiSclMo" defined in clause B.2.1.

4) Default values for TRPDT and RCAT
The M2M Service Provider shall be authorized to provision information to the local SCL the default values to use in clause 9.3.1.5.5 for TRPDT and/or RCAT if they are not provided by the request issuer. These policies shall define which TRPDT value to use as a default value for a given RCAT value or which is the default RCAT value. The resources that are used to provision policies for default TRPDT and RCAT values in case xREM procedures are used, are defined in clause B.2 as part of the resource "etsiSclMo" defined in clause B.2.1.

These types of policies can be provisioned to the NSCL either as application-specific policies (identified by an APP-ID and a SCL-ID) or as D/GSCL-specific policies (identified by a SCL-ID) or as default policies.

The support of policies is mandatory. The policies shall be applied, according to the following:

On the NSCL:

- if present the application-specific policies for the request issuer and the request destination SCL shall be used; otherwise
- if present, the D/GSCL-specific policies for request destination SCL shall be used; otherwise
- if present, default policies shall be used; otherwise
- the request is rejected.

On the D/GSCL the consolidated policies provisioned by the NSCL shall be used as follows:

- if present the application-specific policies for the request issuer shall be used; otherwise
- if present, default policies shall be used; otherwise
- the request is rejected.

The relationship of who is authorized to set policies and how they affect the SAF handling is summarized in Figure 9.50.

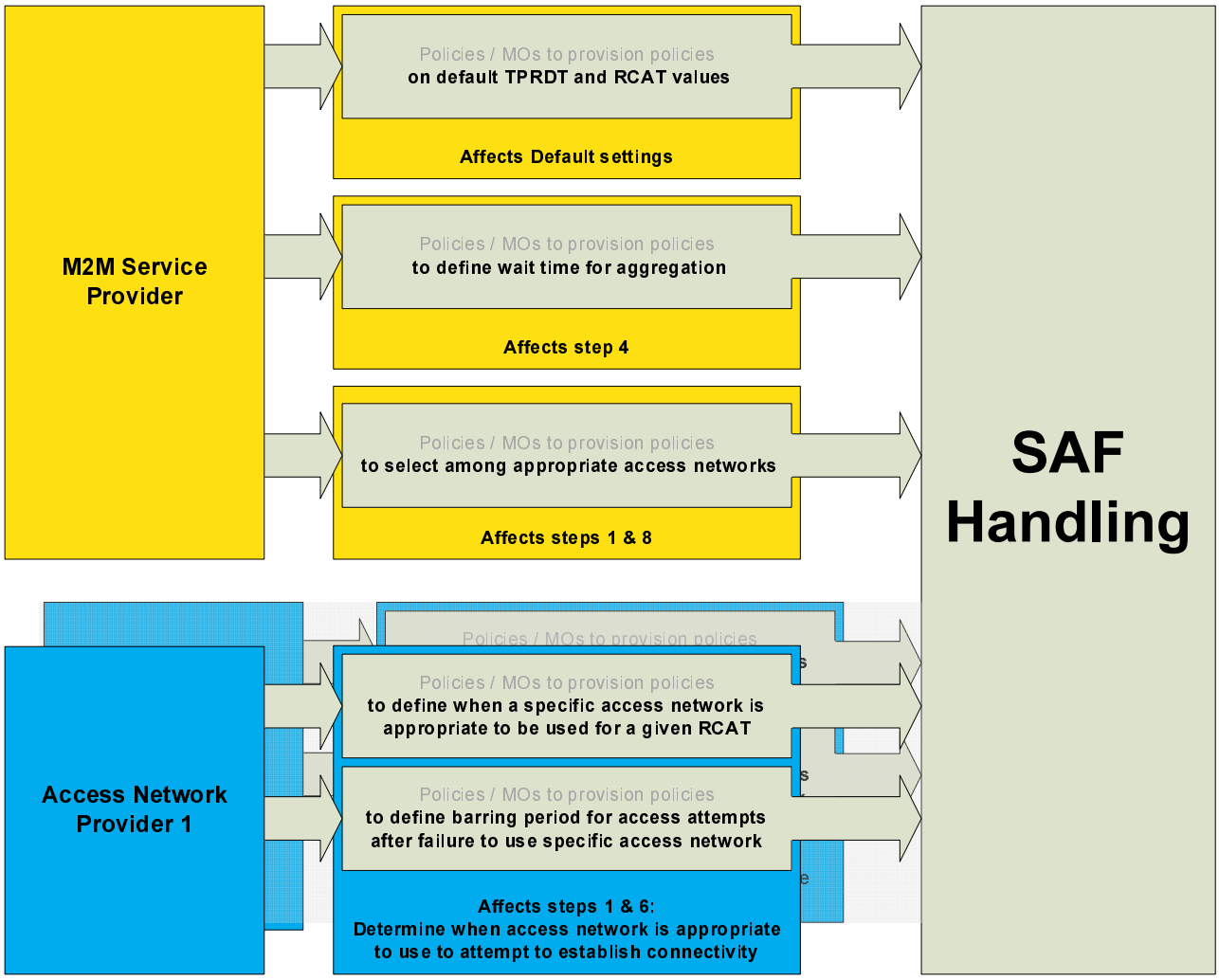


Figure 9.50: Policy types for SAF Handling

9.3.2 Procedure description

9.3.2.1 General

Each procedure described below consists of Create, Retrieve, Update and Delete. For Retrieve and Update procedures the Partial Addressing mechanism as described in clause 9.3.2.29 is applicable. Partial Addressing allows to Create, Retrieve, Update and Delete a single attribute (or part of an attribute). In the remaining of clause 9.3.2 each time a Retrieve and Update procedures is described it is assumed that Partial Addressing is applicable as per clause 9.3.2.29, unless specifically indicated otherwise.

9.3.2.2 Logical sequence of procedures

The following provides the logical sequence of procedures that shall be followed:

- Discovery of SCLs

NOTE: SCL discovery is not covered in the present document.

- SCL management
- Application management

The following procedures may be performed in any sequence once the above procedures have been performed successfully:

- Access rights management
- Container management
- Group management
- Resource discovery
- Collection management
- Subscription management
- Announce/De-announce
- Object Resource Management

When accessing the resources according to the above procedures, the following procedures give some options in the resource interaction:

- Partial Addressing: allows addressing and manipulation of individual attributes or parts of attributes in a resource.
- Long Polling: gives an option for non-server capable clients to still receive asynchronous notifications.

Any procedures that are applicable on the mId reference point shall have established a secure connection as described in clause 8.5 on mId security.

9.3.2.3 Resource name allocation

For procedures that use the Create method to create a resource, there are two cases to be distinguished:

- Issuer does not provide a name: the SCL shall allocate a name and provide it in the Success response, except for the SCL.
- Issuer provides a name of the resource as part of the request: The hosting SCL shall use the suggested name for creating the resource if a resource with that name does not already exist. If the name already exists, the hosting SCL shall choose a name. This procedure is valid for all resources with the exception of the SCL and Application registration; see clauses 9.3.2.6 and 9.3.2.8 for more details.

9.3.2.4 Discovery of <*sclBase*>

It is assumed that the discovery of the <*sclBase*> is performed by means of configuration.

9.3.2.5 SCL collection management

9.3.2.5.1 Introduction

This clause describes different procedures for managing the collection resource *scls* as defined in clause 9.2.3.3. Such type of resource (like any other collection) cannot be created or deleted by means of a request over the reference points: mIa, mId and dIa.

9.3.2.5.2 Retrieve *scls*

This procedure shall be used for getting all attributes of the *scls* collection resource and the list of references to all <*scl*> children resources, for which the Issuer is authorized to discover.

The procedure is described in details in clause 9.3.2.30.2.

9.3.2.5.3 Update *scls*

This procedure shall be used for modifying the attributes in the *scls* collection resource.

The procedure is described in details in clause 9.3.2.30.3.

9.3.2.5.4 Subscribe/Un-Subscribe to *scls*

This procedure shall be used for subscribing and un-subscribing to changes in a *scls* resource.

The procedures are described in details in clause 9.3.2.30.6.

9.3.2.6 SCL management

9.3.2.6.1 Introduction

This clause describes different procedures for managing registrations, de-registrations, updates and retrievals of the registration information associated with SCLs.

9.3.2.6.2 Create <*scl*> (Register SCL)

This procedure shall be used to register an SCL (Issuer-SCL) to another SCL (Hosting-SCL) in order to be able to interact with it. As described in clause 9.2.3.4, when an SCL registers to another SCL, two <*scl*> resources are created, one in the Issuer-SCL and one in the Hosting-SCL storing the information that each SCL needs to know about the other one. The creation of these two resources does not imply a mutual registration, but rather the result of a successful registration of the Issuer-SCL to the Hosting-SCL.

The following provides the list of possible registrations:

- A DSCL registration to a NSCL.
- A GSCL registration to a NSCL.

The SCL-ID of the Issuer-SCL (see clause 7.2.1.3) shall be used as a name for the resource to be created. The SCL-ID and the sclBase of the Issuer-SCL have 1 to 1 relationship. If an Issuer-SCL requests to register more than once to the same Hosting-SCL, the request will be rejected.

The optional Integrity Validation for SCL management procedures are defined in annex C.

The registration procedure consists of the following steps:

Step 1. Issuer-SCL: shall request a registration to the Hosting-SCL by requesting the creation of a new <*scl*> resource using a CREATE verb. The request shall address the <*sclBase*>*/scls* collection of the resource structure as described in clause 9.2.3.4. The Issuer-SCL shall provide a SCL-ID as a name of the new resource that will be created in the Hosting-SCL, it shall also provide the URI of the <*sclBase*> of the Issuer in the "link" attribute and it may also provide the values of the attributes associated to the resource.

Step 2. Hosting SCL: after ensuring that the Issuer-SCL has been authenticated, it shall verify that the SCL-ID does not already exist in the <*sclBase*>*/scls* collection. If it exists, it shall reject the request returning an error. After ensuring the validity of the request and the attributes provided, a new <*scl*> resource structure with the SCL-ID shall be generated. The creation of the SCL resource shall always be allowed if the issuer was successfully authenticated. This is due to the fact that the SCL is not known prior to registration, therefore the accessRights associated to the <*sclBase*>*/scls* collection does not contain a priori the permission for application that is being registered.

Initially, all the sub-resources defined as child of the <*scl*> Resource specified in clause 9.2.3.4 (*containers*, *groups*, *applications*, *accessRights*) shall be created, in addition to initializing the resource attributes with the information initially provided by the issuer SCL. In the case of a successful response, the Receiver shall provide inside the response body the URI of the newly created resource.

Step 3. Issuer SCL: In the case it receives a successful response; it shall create a new local <*scl*> resource, using the SCL-ID of the Hosting SCL. The values of the attributes for the local <*scl*> *resource* shall be assigned using default values or requesting them to the Hosting-SCL. The "expirationTime" attribute, by default, shall be assigned to an infinite timer.

Step 4. Issuer SCL: may request the value of *"searchStrings"* and "*accessRightID*" attributes from the Hosting-SCL by using RETRIEVE on the <*sclBase*> resource of the Hosting SCL. The values of this attributes is stored in the Hosting-SCL under the "searchStrings" and "*accessRightID*" attribute of the <*sclBase*> resource, as indicated in clause 9.2.3.2.

Step 5. Hosting SCL: after ensuring that the Issuer has the access rights to read the attributes of the <sclBase> resource, the Hosting-SCL shall respond to the Issuer providing the values of the requested attributes.

Step 6. Issuer SCL: In the case it receives a successful response; it shall update the values of the attributes of the local <*scl*> resource representing the Hosting SCL.

In order to keep the values of the "searchStrings" and "*accessRightID*" attributes synchronized, the Issuer-SCL may subscribe to changes in the values of these attributes in the <*sclBase*> resource of the Hosting-SCL.

Step 7. Issuer SCL: In case the issuer SCL received an *aPocHandling* attribute in the success response in step 2, it shall set the *aPocHandling* attribute of its local <sclBase> resource to the received value. If no aPocHandling attribute is received the issuer SCL cannot know the value as decided by the NSCL, therefore, it shall set the aPocHandling attribute on its local <sclBase> resource to SHALLOW. The issuer SCL may execute a "retrieve <scl>" procedure to retrieve the *aPocHandling* value from <*scl*> resource created the hosting SCL. If the "retrieve <scl>" procedure succeeds, the issuer SCL may set the aPocHandling attribute on its local <sclBase> resource to the received value.

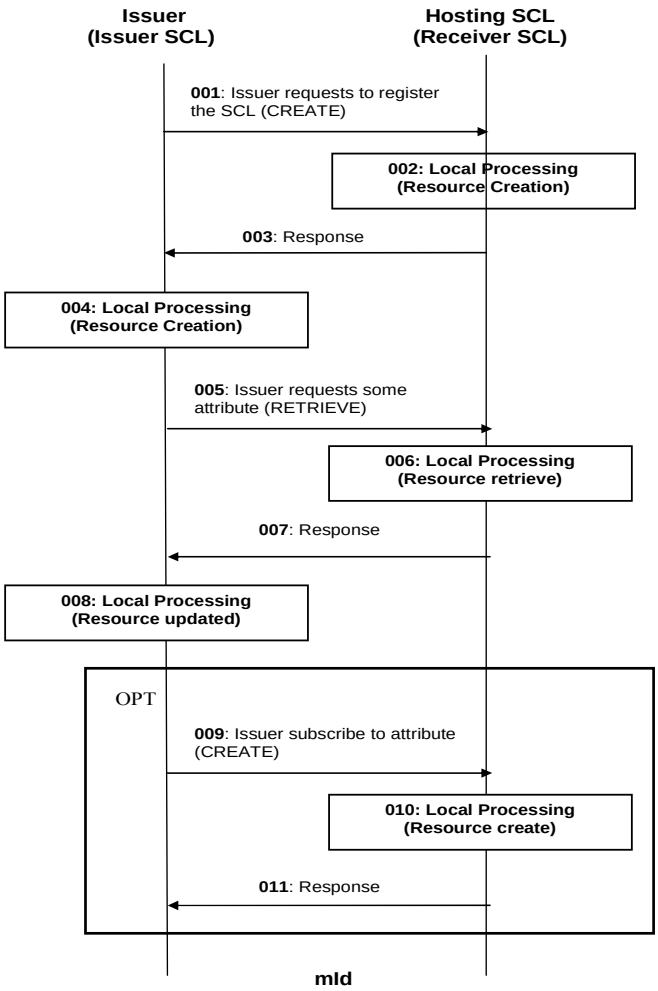


Figure 9.51: Procedures for SCL registration

Step 001: The Issuer SCL shall send a CREATE request to the Hosting SCL. The SCL shall provide the name of the resource as indicated in Step 1 above.

Step 002: The Hosting SCL shall check if the Issuer is authorized to create the resource for the registration. The SCL includes the characteristics of the resource to be created. Characteristics can contain information like the expiration time. More details about the behaviour of the Hosting SCL are provided above in Step 2.

Step 003: The Hosting SCL responds positively to the request.

Step 004: The Issuer SCL shall update its local <sclBase> resource with the received value of the *aPocHandling* attribute or it shall set the value to SHALLOW in case no aPocHandling attribute is received. The issuer SCL shall create a resource representing the Hosting SCL. The Issuer SCL provides some default characteristics for the resource created. More details about the behaviour of the Issuer SCL are provided above in Step 3.

Step 005: The Issuer SCL shall RETRIEVE the <sclBase> resource, in order to read the values for the attribute "searchStrings" and "*accessRightID*" as indicated in Step 5 above.

Step 006: The Hosting SCL shall check if the Issuer is authorized to RETRIEVE the attribute "*searchStrings*" as indicated in Step 6 above in the procedure text.

Step 007: The Hosting SCL responds positively to the request.

Step 008: The Issuer SCL shall UPDATE the local resource representing the Hosting SCL with the values for the attribute "*searchStrings*" and "*accessRightID*" as indicated in Step 7 above.

Step 009: Optionally the Issuer SCL may retrieve the <scl> resource that is created in Step 002. This is only needed in case the response received in step 003 does not contain any resource representation and the issuer SCL is interested in aPocHandling different from SHALLOW (which is the default). The only way for the issuer SCL to discover what aPocHandling is allowed by the NSCL, is to retrieve the <*scl*> resource.

Step 010: The hosting SCL retrieves the <*scl*> resource

Step 011: The Issuer SCL receives the <*scl*> resource and the sets the received *aPocHandling* attribute value on its local <*sclBase*> resource. This value is then later used during the scl retargeting procedures.

Step 012: Optionally the Issuer subscribes to changes of the attributes "*accessRightID*" and "*searchStrings*" for keeping the values synchronized. The Issuer provides some characteristics for the subscription.

Step 013: The Hosting SCL shall check if the Issuer is authorized to create the resource for the subscription. The SCL includes the characteristics of the resource to be created. Characteristics can contain information like the expiration time (see clause 9.3.2.29.7 for more details).

Step 014: The Hosting SCL responds positively to the request.

List of main procedure specific exceptions:

Step 002: The provided name already exists. The Hosting SCL responds with an error.

Step 002: The provided characteristics are not acceptable by the Hosting SCL. The Hosting SCL responds with an error. The flow start all over with step 001, if the Issuer SCL provides new characteristics, otherwise the flow terminates here.

Step 006: The Issuer is not authorized to retrieve the attributes. The Hosting SCL responds with an error.

Step 009: The provided characteristics are not acceptable by the Hosting SCL. The Hosting SCL responds with an error.

Step 010: The Issuer is not authorized to create the resource The Hosting SCL responds with an error.

Step 010: The provided characteristics are not acceptable by the Hosting SCL. The Hosting SCL responds with an error.

9.3.2.6.3 Retrieve <*scl*>

This procedure shall be used for retrieving any of the existing information stored in any of the attributes that compose an SCL resource. This procedure shall work in the same manner as the retrieval of any other resource. Alternatively, the issuer can request to retrieve only a specific attribute or part of an attribute, as described in partial addressing, see clause 9.3.2.29.

Issuer: requests to retrieve any of the attributes of the existing SCL resource by using RETRIEVE. The request shall address the specific <SCL-ID> resource of the Hosting SCL.

The request addresses an <*scl*> resource as defined in clause 9.2.3.4 (resource structure).

Hosting SCL: after it verified the existence of the addressed <*scl*> resource and the permissions for retrieving the resource, the SCL shall return the requested information.

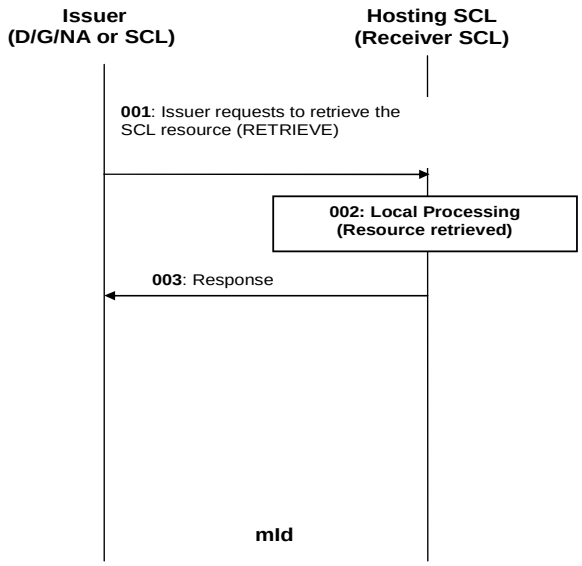


Figure 9.52: Procedures for <*scl*> retrieve

Step 001: The Issuer requests to retrieve the <*scl*> resource from the Hosting SCL.

Step 002: The Hosting SCL shall check if the Issuer is authorized to retrieve the <*scl*> resource. More details about the behaviour of the Hosting SCL are provided above.

Step 003: The Hosting SCL responds positively to the request.

List of main procedure specific exceptions:

Step 002: The Issuer is not authorized to retrieve the resource. The Hosting SCL responds with an error.

9.3.2.6.4 Update <*scl*>

This procedure shall be used for updating any of the existing SCL registration attributes stored in the Hosting SCL. Especially important is the "expirationTime", since a failure in refreshing this attribute may result in the deletion of the resource. Alternatively, the issuer can request to update only a specific attribute or part of an attribute, as described in partial addressing, see clause 9.3.2.29.

Issuer: requests to update the <*scl*> resource as defined in clause 9.2.3.4 (resource structure). The issuer shall send new (proposed) values for all mandatory read-write attributes and may send values for the optional read-write attributes.

Hosting SCL: after it verified the existence of the addressed <*scl*> resource, the validity of the provided attributes and the permissions to modify them, the Hosting SCL shall update the attributes and shall respond to the issuer with the appropriate generic responses as indicated in clause 9.3.1.1.

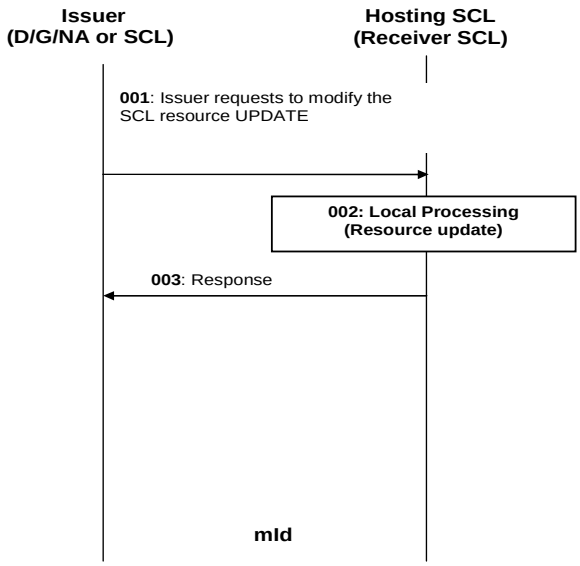


Figure 9.53: Procedures for SCL update

Step 001: The Issuer requests the Hosting SCL to modify a <*scl*> resource. The Issuer has three options for updating the resource as indicated in the text above. The Issuer can therefore use an UPDATE.

Step 002: The Hosting SCL shall check if the Issuer is authorized to perform the modification to the <*scl*> resource. The Hosting SCL shall perform the changes to the resource. More details about the behaviour of the Hosting SCL are provided above.

Step 003: The Hosting SCL responds positively to the request.

List of main procedure specific exceptions:

Step 002: The Issuer is not authorized to modify (update, create or delete) the resource. The Hosting SCL responds with an error.

9.3.2.6.5 Delete <*scl*> (De-Register SCL)

This procedure shall be used by the Issuer-SCL to de-register from the Hosting-SCL. When an SCL registers to another SCL, two resources are created, one in the Issuer SCL and another in the Hosting SCL. The de-registration process shall consist in the deletion of both resources previously created.

Step 1. Issuer: requests a de-registration from the Hosting SCL by requesting the deletion of the <*scl*> resource named as SCL-ID using DELETE.

Step 2. Hosting SCL: after it verified the existence of the addressed SCL-ID resource and the validity of permissions for deleting the resource, then the Hosting SCL shall remove the resource from the repository and shall respond to the issuer with the appropriate generic responses as indicated in clause 9.3.1.1.

NOTE: The SCL does not get notified by a de-registration performed by other entities (e.g. other Applications or SCL) which are authorized to perform such action, unless specifically requested to be notified by means of subscriptions.

Step 3. Issuer: After receiving the successful response, the Issuer-SCL shall delete the <*scl*> resource representing the registered-to SCL (Hosting SCL).

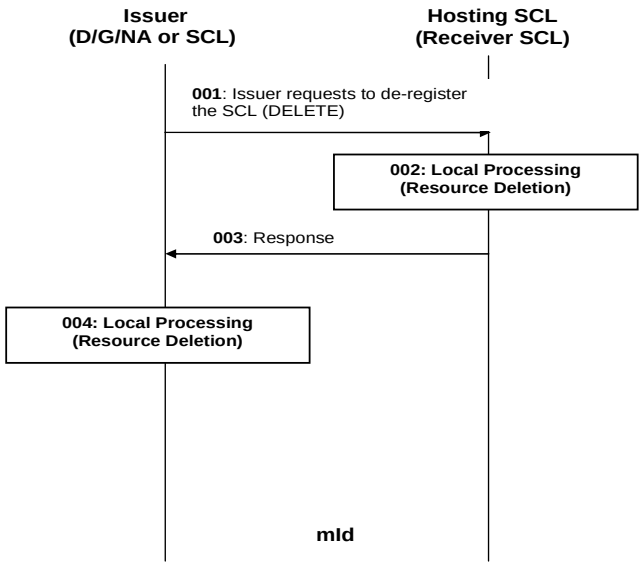


Figure 9.54: Procedures for SCL de-registration

Step 001: The Issuer shall send a DELETE request to the Hosting SCL, see step 1 above for more details.

Step 002: The Hosting SCL shall check if the Issuer is authorized to delete the resource for the registration. The SCL shall remove the resource. More details about the behaviour of the Hosting SCL are provided above in step 2.

Step 003: The Hosting SCL responds positively to the request.

Step 004: The Issuer SCL shall delete the local resource representing the Hosting SCL.

List of main procedure specific exceptions:

Step 002: The Issuer is not authorized to delete the resource. The Hosting SCL responds with an error and the flow terminates here. Step 004 is not performed.

9.3.2.6.6 Subscribe/Un-Subscribe to <*scl*>

This procedure shall be used for subscribing and un-subscribing to changes in a <*scl*> resource and managing the subscription itself. The procedures are described in details in clause 9.3.2.19.

9.3.2.7 Applications collection management

9.3.2.7.1 Introduction

This clause describes different procedures for managing the collection resource *applications* as defined in clause 9.2.3.5. Such type of resource (like any other collection) cannot be created or deleted by means of a request over the reference points: mIa, mId and dIa.

9.3.2.7.2 Retrieve applications

This procedure shall be used for getting all attributes of the *applications* collection resource and the list of references to all <*application*> children resources, for which the Issuer is authorized to discover.

The procedure is described in details in clause 9.3.2.30.2.

9.3.2.7.3 Update applications

This procedure shall be used for modifying all or some of the attributes defined in the *applications* collection resource.

The procedure is described in details in clause 9.3.2.30.3.

9.3.2.7.4 Subscribe/Un-Subscribe to *applications*

This procedure shall be used for subscribing and un-subscribing to changes in an *applications* collection resource.

The procedure is described in details in clause 9.3.2.30.6.

9.3.2.8 Application management

9.3.2.8.1 Introduction

This clause describes different procedures for managing registrations, de-registrations, updates and retrievals of the registration information associated with an application. It is assumed that prior to any of the procedures described below; the Application that performs the request has been properly Authenticated and Authorized.

NOTE: The mechanisms for authenticating and authorizing the Application are out of scope of the present document.

9.3.2.8.2 Create <*application*> (Register Application)

This procedure shall be used for registering and implicitly creating an <*application*> resource on the Local SCL, which is also the Hosting SCL. The registration is done via the mIa or dIa.

Issuer: requests to register an <*application*> by using a CREATE. The Issuer shall:

1) either provide an APP-ID. The Identifier shall be globally unique; how the uniqueness is achieved is out of scope; or

2) no identifier at all.

The request shall address the resource collection <*sclBase*>*/applications/*. In addition the issuer may request to announce the resource in several SCLs, for more details on the procedure for announcing a resource, see clause 9.3.2.28.

Hosting SCL: after ensuring that the Issuer has been authenticated, it shall:

1) Verify that the provided identifier does not already exist in the *applications* collection resource. This is only needed in case that the Issuer has provided the identifier in the request. If it exists in the *applications* collection, the Hosting SCL shall reject the request returning an error.

2) Create an identifier, APP-ID and assign it to the Issuer Application. The Identifier shall be globally unique; how the uniqueness is achieved is out of scope.

Then the Hosting SCL shall validate the received request and it shall create an <*application*> resource with the specified attributes.

The creation of the <*application*> resource shall always be allowed if the issuer has been authenticated. This is due to the fact that the application is not known prior to registration, therefore the accessRights associated to the <*sclBase*>*/applications* collection does not contain a priori the permission for application that is being registered.

The Hosting SCL may also provide default values for not provided optional attributes; these are inferred from the SCL policies. For example, the Hosting SCL may reduce the expiration time suggested by the issuer.

The hosting SCL shall enforce the expiration time of the <*application*> resource. The issuer or any other party allowed to modify the application resource may extend the lifetime of the resource by modifying the expiration time attribute, clause 9.2.3.6.

When the expiration time is reached, the Hosting SCL shall remove the <*application*>*n* resource. This means that also all child resources shall be deleted.

After creation of the <*application*> resource, the Hosting SCL shall return a generic response as indicated in clause 9.3.1.1. A successful response shall include the URI of newly created <*application*> resource. The URI shall contain the Identifier provided by the issuer and the URI is globally unique.

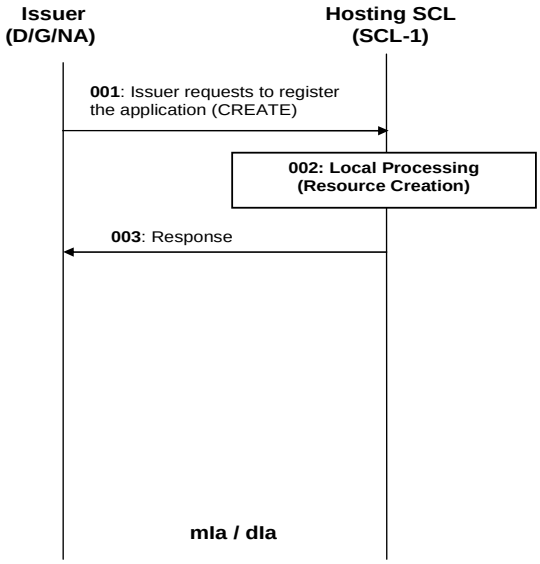


Figure 9.55: Procedures for application registration

Step 001: The Issuer application shall send a CREATE request to its Local SCL(=Hosting SCL). The Application may provide the name of the resource.

Step 002: The Hosting SCL shall check if the Issuer is authorized to create the resource for the registration. The SCL includes the characteristics of the resource to be created. Characteristics can contain information like the expiration time. More details about the behaviour of the Hosting SCL are provided above.

Step 003: The Hosting SCL responds positively to the request.

List of main procedure specific exceptions:

Step 002: The provided characteristics are not acceptable to the Hosting SCL. The Hosting SCL responds with an error.

9.3.2.8.3 Retrieve <application>

This procedure shall be used for retrieving the content of an <*application*> resource. It can either be the whole <*application*> resource representation or a specific attribute or part of an attribute. The content of the application consists of all attributes and the URIs of the child resources as described in clause 9.2.3.6.

Issuer: requests to retrieve the information of an <*application*> resource by using the RETRIEVE. The request may address the specific <*APP-ID*> resource of the Hosting SCL.

The request addresses an <*application*> resource as defined in clause 9.2.3.6.

Hosting SCL: shall validate the received request. Retrieval shall only be allowed if the issuer is authorized to retrieve the <*application*> resource, according to the accessRight permission defined for the resource itself. The Hosting SCL shall provide the requested information. If the requestor is not the application itself, then the *aPoC* attribute shall be removed from the resource representation before returning it.

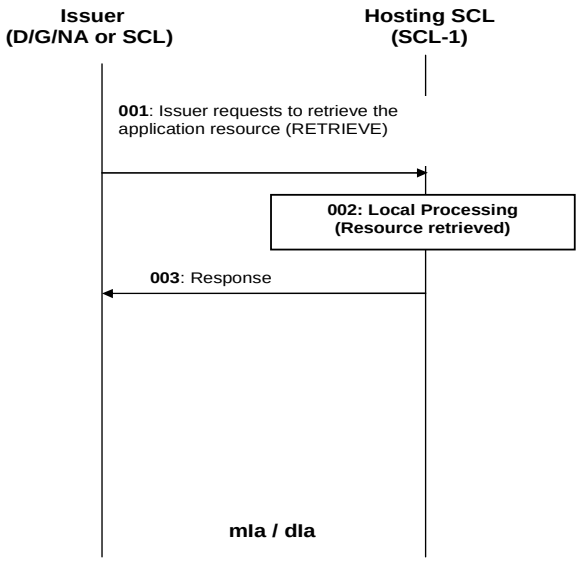


Figure 9.56: Procedures for application retrieve

Step 001: The Issuer requests to the Hosting SCL to retrieve an application resource.

Step 002: The Hosting SCL shall check if the Issuer is authorized to retrieve the <*application*> resource. More details about the behaviour of the Hosting SCL are provided above.

Step 003: The Hosting SCL responds positively to the request.

List of main procedure specific exceptions:

Step 002: The Issuer is not authorized to retrieve the resource. The Hosting SCL responds with an error.

9.3.2.8.4 Update <application>

This procedure shall be used for updating an <*application*> resource. Alternatively, the issuer can request to update only a specific attribute or part of an attribute, as described in partial addressing, see clause 9.3.2.29.

Issuer: requests to update an <*application*> resource as defined in clause 9.2.3.6 (resource structure), by means of the UPDATE. The issuer shall send new (proposed) values for all mandatory read-write attributes and may send values for the optional read-write attributes.

Hosting SCL: shall validate the received request. Update shall only be allowed if the issuer is authorized to write the <*application*> resource, according to the accessRight permissions defined for the <*application*> resource itself. The hosting SCL shall then modify the resource representation according to the request. The hosting SCL may modify some of the attributes, e.g. expiration time, in accordance with SCL policies. In case the issuer requested to delete one of the optional attributes, the hosting SCL shall provide default values for attributes that are required by the SCL.

A modification of the <*application*> resource may trigger an announce or de-announce the application, see clause 9.3.2.28 for more details on the specific procedure.

The hosting SCL returns a generic response according to clause 9.3.1.1.

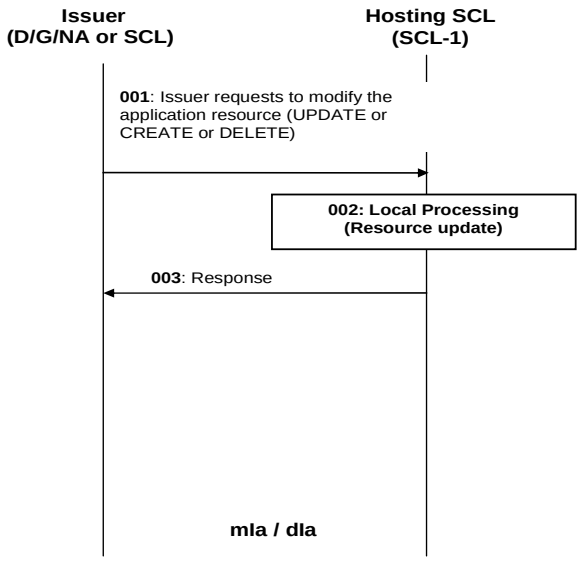


Figure 9.57: Procedures for <*application*> update

Step 001: The Issuer requests the Hosting SCL to modify an <*application*> resource. The Issuer has three options for updating the resource as indicated in the text above. The Issuer can therefore use an UPDATE or a CREATE or a DELETE.

Step 002: The Hosting SCL shall check if the Issuer is authorized to perform the modification to the <*application*> resource. The SCL shall perform the changes to the resource. More details about the behaviour of the Hosting SCL are provided above.

Step 003: The Hosting SCL responds positively to the request.

List of main procedure specific exceptions:

Step 002: The Issuer is not authorized to modify (update, create or delete) the resource. The Hosting SCL responds with an error.

9.3.2.8.5 Delete <*application*> (De-register Application)

This procedure shall be used for de-registering and deleting all information related to the application.

Issuer: requests to delete the application resource by using a DELETE. The issuer shall be either an application or an SCL.

Hosting SCL: shall validate the received request. Delete shall only be allowed if the issuer is authorized to delete the <*application*> resource, according to the accessRights defined for the <*application*> resource where the request is targeted.

The hosting SCL shall de-announce the application from all SCLs where the application was previously announced. For more details on the de-announcing procedure, see clause 9.3.2.28. Then the hosting SCL deletes the addressed resource, including all the children resources.

The hosting SCL returns a generic response according to clause 9.3.1.1.

NOTE: The Application does not get notified by a de-registration performed by other entities (e.g. other Applications or SCL) which are authorized to perform such action, unless specifically requested to be notified by means of subscriptions.

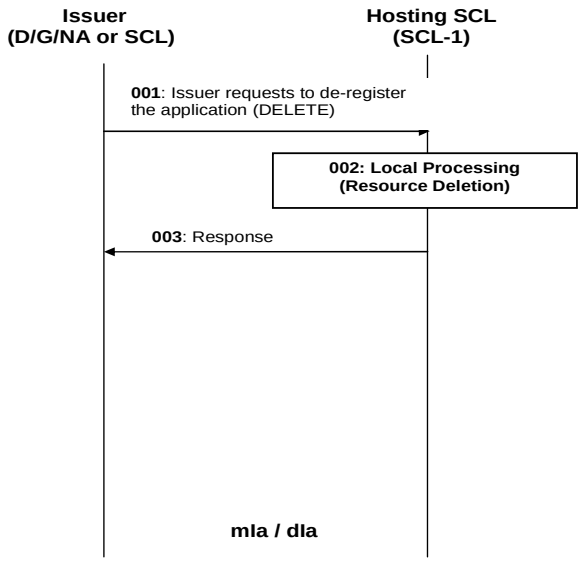


Figure 9.58: Procedures for application de-registration

Step 001: The Issuer shall send a DELETE request to its Local SCL (= Hosting SCL).

Step 002: The Hosting SCL shall check if the Issuer is authorized to delete the resource for the registration. The SCL shall remove the resource. More details about the behaviour of the Hosting SCL are provided above.

Step 003: The Hosting SCL responds positively to the request.

List of main procedure specific exceptions:

Step 002: The Issuer is not authorized to delete the resource. The Hosting SCL responds with an error.

9.3.2.8.6 Subscribe/Un-Subscribe to <*application*>

This procedure shall be used for subscribing and un-subscribing to changes in an <*application*> resource and managing the subscription itself. The procedures are described in details in clause 9.3.2.19.

9.3.2.8.7 Create <applicationAnnc> (Announce/de-announce an <application>)

This procedure shall be used for announcing and/or de-announcing an <*application*> resource to/from another SCL. The announce procedure creates a resource <*applicationAnnc*> in the Announce-To SCL.

The procedure is described in details in clause 9.3.2.28.

9.3.2.9 *accessRights* collection management

9.3.2.9.1 Introduction

This clause describes different procedures for managing the retrieval and updates of information associated with an *accessRights* resource as defined in clause 9.2.3.8. Such type of resource (like any other collection) cannot be created or deleted by means of a request over the reference points: mIa, mId and dIa.

9.3.2.9.2 Retrieve accessRights

This procedure shall be used for getting all attributes of the *accessRights* collection resource and the references to children resources, for which the Issuer is authorized to discover.

The procedure is described in details in clause 9.3.2.29.2.

9.3.2.9.3 Update accessRights

This procedure shall be used for modifying the attributes in the *accessRights* collection resource.

The procedure is described in details in clause 9.3.2.30.3.

9.3.2.9.4 Subscribe/Un-Subscribe *accessRights*

These procedures shall be used for subscribing and un-subscribing to changes in an *accessRights* resource.

The procedures are described in details in clause 9.3.2.30.6.

9.3.2.10 Access Right management

9.3.2.10.1 Introduction

This clause describes different procedures for managing the creation, retrieval, update and deletion of information associated with an <*accessRight*> resource.

9.3.2.10.2 Create <accessRight>

This procedure shall be used to create an <*accessRight*> resource. Such an <*accessRight*> resource can later be associated with other resources in order to control access to those resources for security reasons. The issuer may optionally suggest a name for the resource or it can let the hosting SCL provide the name.

Issuer: shall request to create a new <*accessRight*> resource using the CREATE verb.
The request shall address an existing *accessRights* collection resource as described in clause 9.2.3.8.
The request may contain a suggested name for the resource to be created.
The request shall provide selfPermissions and may provide permissions, expirationTime and searchStrings attributes.

The issuer can be an application or an SCL.

Hosting SCL: shall validate the provided attributes. In particular the hosting SCL shall validate whether the URIs of the permissionHolders in the permissions attribute can potentially refer to an <*sclBase*> resource, an <*application*> resource or a <*group*> resource. Specifically, the URIs used as permissionHolders shall not refer to <*scl*> resources, <*applicationAnnc*> resources, to <*groupAnnc*> resources or to entities that can not represent an <*scl*>, <*application*> or <*group*> resource (e.g. `<localSclBase>/containers/<someContainer>`).

Furthermore, the hosting SCL shall check that the issuer has the permissions to create a new <*accessRight*> resource (i.e. if the issuer has CREATE_CHILD permission for the addressed accessRights collection resource).
Then a new <*accessRight*> resource shall be created in the hosting SCL repository taking into account [resource naming]. The hosting SCL may modify some of the provided attributes before creating the <*accessRight*> resource.
Then the SCL shall respond to the issuer with the appropriate generic responses as indicated in clause 9.3.1.1, it shall also provide in the response the resource URI of the created resource.

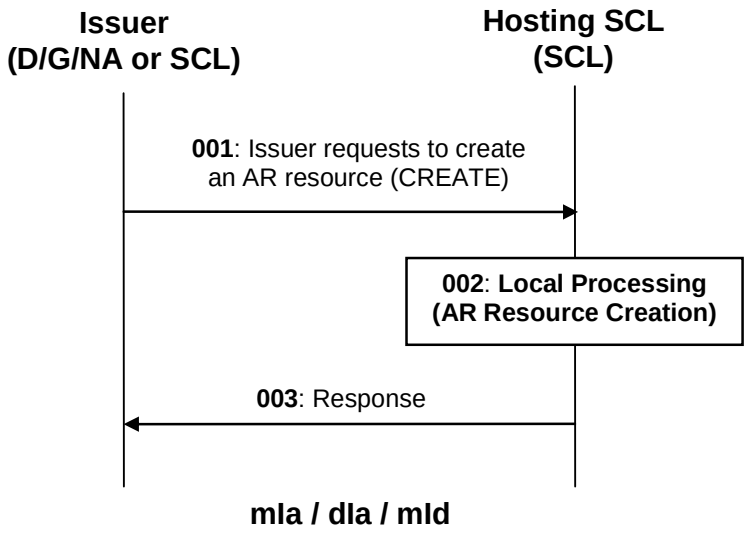


Figure 9.59: Message flow of procedure to create Access Rights resources

Step 001: To create an <*accessRight*> resource, the application or SCL shall issue a create request to the Hosting SCL where the resource shall be created. The request shall include authentication data and may include the attributes of the resource to be created, like the expiration time.

Step 002: If the creation is allowed by Hosting SCL the resource shall be created.

Step 003: A success result shall be returned as a response and returned to the issuer. The result shall indicate the URI of the created resource.

List of procedure specific exceptions:

Step 002: The requesting application or SCL is not registered.

Step 002: The requesting application or SCL does not have the authorization to create this resource.

Step 002: The provided attributes are not acceptable to the Hosting SCL.

9.3.2.10.3 Retrieve <accessRight>

This procedure shall be used for retrieving the representation of an existing <*accessRight*> resource, i.e. by retrieving the values of all attributes of its structure and references to all its child resources.

Issuer: shall request to retrieve the information of an <*accessRight*> resource by using the RETRIEVE verb. The request shall address the URI of the <*accessRight*> resource.

The issuer can be an application or an SCL.

Hosting SCL: shall check the existence of the addressed <*accessRight*> resource and shall check if according to the selfPermissions of the addressed <*accessRight*> resource the issuer has the permission to retrieve the resource. Then the SCL shall read the values of all attributes belonging to the addressed resource structure and the references of all sub-resources and it shall build a representation of these. This representation is included in an appropriate response that is sent to the issuer according to clause 9.3.1.2.

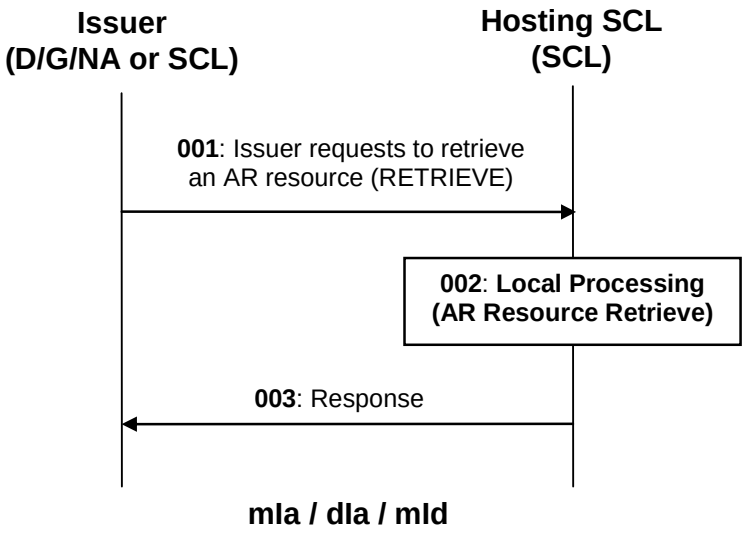


Figure 9.60: Message flow of procedure to retrieve Access Rights resources

Step 001: To retrieve an <*accessRight*> resource, the application or SCL shall issue a retrieve request addressing the resource in the Hosting SCL.

Step 002: If the retrieve operation is allowed by Hosting SCL the representation of the resource shall be read for building the response.

Step 003: A success result shall be returned to the issuer as a response in addition to the representation of the resource.

List of procedure specific exceptions:

Step 002: The requesting application or SCL is not registered.

Step 002: The addressed resource does not exists.

Step 002: The requesting application or SCL does not have the authorization to read this resource.

9.3.2.10.4 Update <accessRight>

This procedure shall be used to update/modify an existing <*accessRight*> resource.

Issuer: may request to update the attributes of an existing <*accessRight*> resource, by using UPDATE verb.

The request shall address a specific <*accessRight*> resource in an SCL to replace the entire *accessRight* resource representation. The issuer provides attribute values similar to the create procedure. The Issuer may send new (proposed) values for all mandatory read-write attributes and may send values for the optional read-write attributes.
The issuer can be an application or an SCL.

Hosting SCL: shall check the existence of the addressed <*accessRight*> resource and shall check if according to the selfPermissions of the addressed <*accessRight*> resource, the issuer has the permission to update the resource. Upon successful validation the hosting SCL shall overwrite the attributes of the <*accessRight*> resource with the provided values, but the hosting SCL may modify some of the provided attribute values as in the create procedure. The hosting SCL shall respond to the issuer with the appropriate generic responses as indicated in clause 9.3.1.1.

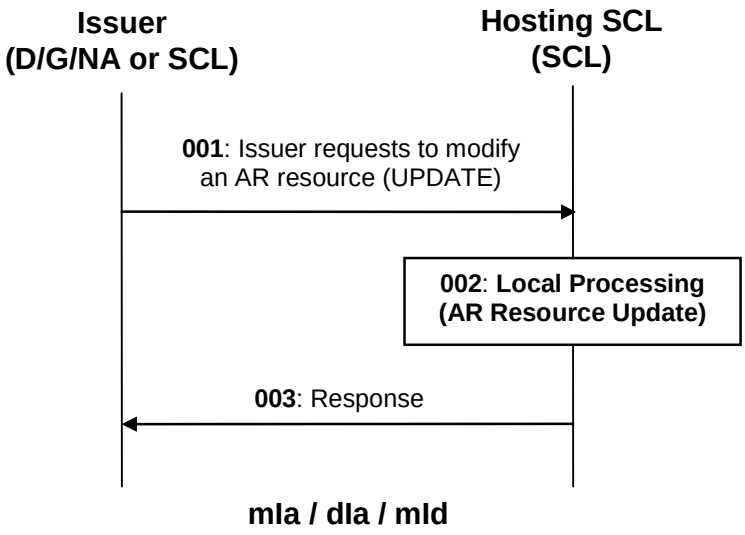


Figure 9.61: Message flow of procedure to modify Access Rights resources

Step 001: To modify an <*accessRight*> resource, the application or SCL shall issue an update request to Hosting SCL. The request shall include the new resource representation for the resource.

Step 002: If the update is allowed by Hosting SCL the resource shall be updated with the provided resource representation, suitably modified, if needed.

Step 003: A success result shall be returned as a response to the issuer.

List of procedure specific exceptions:

Step 002: The requesting application or SCL is not registered.

Step 002: The addressed resource does not exists.

Step 002: The requesting application or SCL does not have the authorization to update this resource.

Step 002: The provided attributes are not acceptable to the Hosting SCL.

9.3.2.10.5 Delete <accessRight>

This procedure shall be used to delete an existing <*accessRight*> resource.

Issuer: shall request to delete an existing <*accessRight*> resource by using a DELETE verb. The request shall address an <*accessRight*> resource.

The issuer can be an application or an SCL.

Hosting SCL: shall check the existence of the addressed <*accessRight*> resource and shall check if according to the selfPermissions of the addressed <*accessRight*> resource the issuer has the permission to delete the resource. Then the SCL shall remove it from its repository and shall respond to the issuer with the appropriate generic responses as indicated in clause 9.3.1.1.

Once the access right resource has been deleted, all other resources referencing to the now non-existing URI in their *accessRightID* attribute shall automatically revert to a default or system permissions.

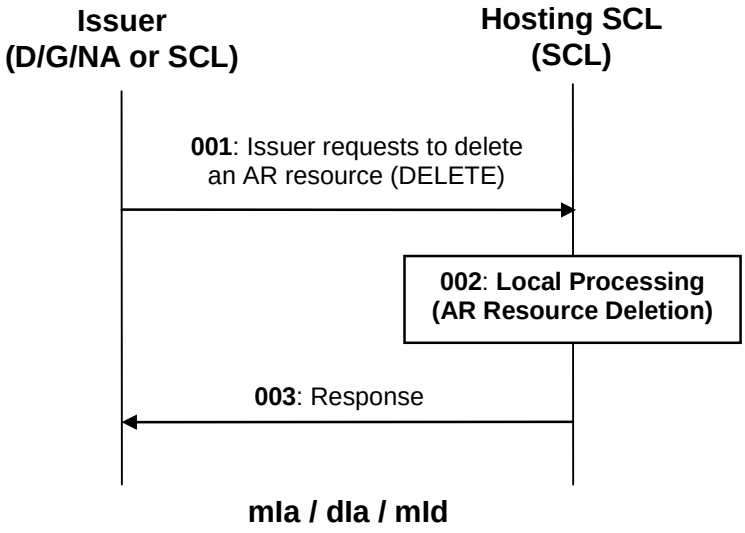


Figure 9.62: Message flow of procedure to delete Access Rights resources

Step 001: To delete an <*accessRight*> resource, the application or SCL shall issue a delete request to the URI of the resource to be deleted in the Hosting SCL.

Step 002: If the deletion is allowed by Hosting SCL the resource shall be deleted.

Step 003: A success result shall be returned as a response to the issuer.

List of procedure specific exceptions:

Step 002: The requesting application or SCL is not registered.

Step 002: The addressed resource does not exists.

Step 002: The requesting application or SCL does not have the authorization to delete this resource.

9.3.2.10.6 Subscribe/Un-subscribe to <*accessRight*>

These procedures are used for subscribing or un-subscribing for changes in an <*accessRight*> resource.

The procedures are described in details in clause 9.3.2.19.

9.3.2.10.7 Create <*accessRightAnnc*> (Announce/de-announce an <accessRight>*)*

These procedures shall be used for announcing and/or de-announcing an <*accessRight*> resource to/from another SCL.

These procedures manipulate (create/delete) a resource <*accessRightAnnc*> in the Announce-To SCL.

The procedure is described in details in clause 9.3.2.28.

9.3.2.11 Container Collection management

9.3.2.11.1 Introduction

This clause describes different procedures for managing the retrieval and update of information associated with a *containers* collection resource. Such type of resource (like any other collection) cannot be created or deleted by means of a request over the reference points: mIa, mId and dIa.

9.3.2.11.2 Retrieve containers

This procedure shall be used to retrieving the representation of a *containers* collection resource. The representation includes the values of all the attributes and the references to the children resources, for which the Issuer is authorized to discover.

The issuer can be an application or an SCL.

The procedure is described in details in clause 9.3.2.30.2.

9.3.2.11.3 Update containers

This procedure shall be used to modify the attributes in a *containers* collection resource.

The issuer can be an application or an SCL.

The procedure is described in details in clause 9.3.2.30.3.

9.3.2.11.4 Subscribe/un-subscribe to *containers*

These procedures shall be used for subscribing or un-subscribing to changes in a *containers* collection resource. For example, subscriptions can be used to be notified when <*container*> resources are added to or deleted from the collection resource.

The procedures are described in details in clause 9.3.2.30.6.

9.3.2.12 Container management

9.3.2.12.1 Introduction

This clause describes different procedures for managing the creation, retrieval, updating and deletion of information associated with a <*container*> resource.

9.3.2.12.2 Create <container>

This procedure shall be used to create a <*container*> resource as a child of the *containers* collection resource. The <*container*> resource is used to exchange information (represented as <*contentInstance*> resources) between one or more producers and one or more consumers.

Issuer: shall request to create a <*container*> resource using the CREATE verb. The request shall address a *containers* collection resource of an SCL as defined in clause 9.2.3.11. There are different *containers* collection resources that can be used depending, primarily, on the lifecycle requirements of the created <*container*> resource. The choice of the *containers* resource collection may also depend on other criteria such as accessRight and application (DA, GA and NA) or SCL logic.

The issuer can be an application or an SCL.

The following lists the different resources and the corresponding lifecycle requirements:

- `<sclBase>/containers`
  The <*container*> resource has a lifecycle independent of any application or remote SCL. The <*container*> resource shall only be removed when it is explicitly requested to be removed.

- `<sclBase>/applications/<application>/containers`
  The <*container*> resource has a lifecycle dependent of a locally registered application that corresponds to the <*application*> resource. The <*container*> resource shall be removed when the <*application*> resource is removed, i.e. when the application is deregistered.

- `<sclBase>/scls/<scl>/containers`
  The <*container*> resource has a lifecycle dependent on a remote SCL that corresponds with the <*scl*> resource. The <*container*> resource shall be removed when the <*scl*> resource is removed, i.e. when the remote SCL is deregistered.

- `<sclBase>/scls/<scl>/applications/<applicationAnnc>/containers`
  The <*container*> resource has a lifecycle dependent on an announced application corresponding to the <*applicationAnnc*> resource. The <*container*> resource shall be removed when the <*applicationAnnc*> resource is removed, i.e. when the corresponding application is de-announced.

The issuer may also include the following information in the request; a resourceName and an expirationTime. It may also propose some policy related restrictions enforced by the hosting SCL, e.g. the maximum number of instances the <*container*> can contain, the maximum age of instances in the <*container*> and the maximum total byte size of all instances combined. The issuer may also include a list of <*sclBase*> URIs to which the created <*container*> shall be announced (see clause 9.3.2.28).

Hosting SCL: shall validate the received request and shall create a <*container*> resource with the specified attributes and the specified resourceName, taking into account [resourcename creation]. Creation shall only be allowed if the issuer is authorized to create a child resource according to the accessRight defined for the *containers* collection where the request is targeted. The hosting SCL may modify some of the attributes, e.g. due to internal policies. For example, it might change the expirationTime suggested by the issuer, or it may restrict the maximum size of the <*container*>. If no *expirationTime* was suggested by the issuer, then the hosting SCL shall set the expiration time to infinite for all created <*container*> resources in the following collections:
`<sclBase>/applications/<application>/containers`,
`<sclBase>/scls/<scl>/containers` or
`<sclBase>/scls/<scl>/applications/<application>/containers`,
since <*container*> resources at these locations are already bound by the lifetime of their parent.

The request also may include a list of <*sclBase*> resource URIs to which the <*container*> resource shall be announced. See clause 9.3.2.28 on how the hosting SCL handles such a request.

NOTE: However, announcing is only to be allowed for locally created <*container*> resources, i.e. residing in the `<sclBase>/containers` and `<sclBase>/applications/<application>/containers` collections.

In addition to creating the <*container*> resource, the hosting SCL shall create the sub-resources of the container, i.e. the *subscriptions*, the *contentInstances* resources. The accessRightID of the *contentInstances* resource shall be initially set to the same accessRightID of the parent.

After creation of the <*container*> resource and its child resources, the hosting SCL shall return a generic response as indicated in clause 9.3.1.1.

The success response shall include the URI of the created <*container*> resource.

The hosting SCL shall enforce the expiration time of the <*container*> resource if it is not infinite. The issuer or any other party allowed to modify the <*container*> resource may extend the lifetime of the <*container*> resource by modifying the expiration time attribute, see clause 9.2.3.12. If the expiration time is exceeded then the resource and all its descendant resources will be removed.

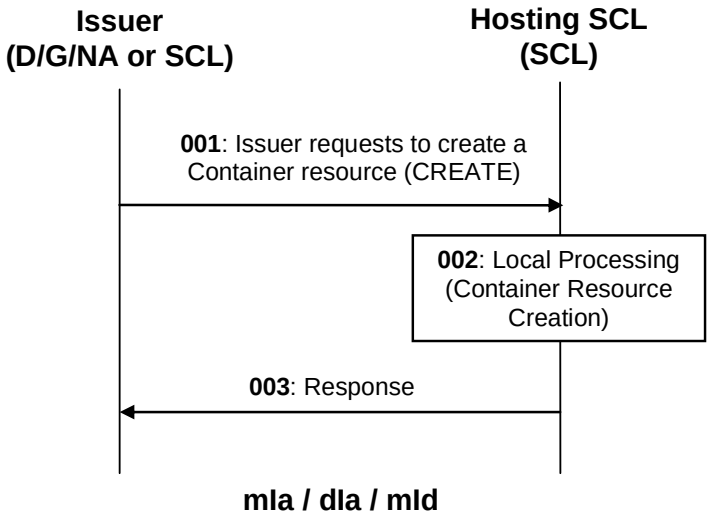


Figure 9.63: Message flow of procedure to create <*container*> resources

Step 001: To create a <*container*> resource, the issuer shall issue a CREATE request.

Step 002: If the creation is allowed by Hosting SCL, the <*container*> resource shall be created.

Step 003: A result shall be returned as a response to the issuer.

List of main procedure specific exceptions:

Step 002: The requesting application or SCL is not registered.

Step 002: The requesting application or SCL does not have the authorization to create this resource.

Step 002: The provided attributes are not accepted to the Hosting SCL.

9.3.2.12.3 Retrieve <container>

This procedure shall be used to retrieve the attributes of a <*container*> and the references of its direct child resources.

Issuer: shall request to retrieve the information of a <*container*> resource, using the RETRIEVE verb. The request shall address an existing <*container*> resource of an SCL as defined in clause 9.2.3.12.

The issuer can be an application or an SCL.

Hosting SCL: shall validate the received request. Retrieval shall only be allowed if the issuer is authorized to retrieve the <*container*> resource, according to the accessRight defined for the <*container*> resource where the request is targeted. The hosting SCL returns a representation of the <*container*> resource, with all the attributes defined in the <*container*> resource and URIs of all the child resources. The hosting SCL shall return a generic response as indicated in clause 9.3.1.1.

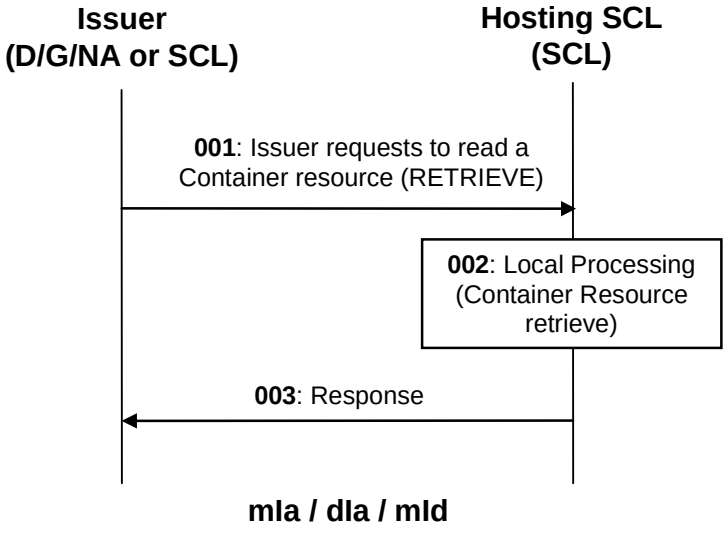


Figure 9.64: Message flow of procedure to read <*container*> resources

Step 001: To read a <*container*> resource, the issuer shall issue a RETRIEVE request.

Step 002: If the retrieve operation is allowed by Hosting SCL, the requested <*container*> resource shall be retrieved.

Step 003: A result shall be returned as a response to the issuer, containing the retrieved information.

List of main procedure specific exceptions:

Step 002: The requesting application or SCL is not registered.

Step 002: The addressed resource does not exists.

Step 002: The requesting application or SCL does not have the authorization to read this resource.

9.3.2.12.4 Update <container>

This procedure shall be used to modify the attributes defined in the <*container*> resource.

A specific usage of this procedure could be to extend the lifetime of a <*container*>resource by updating the expiration time attribute of a <*container*>resource. The <*container*> resource will be removed by the hosting SCL when the expiration time is exceeded. This means that the <*container*> resource lifetime has to be extended explicitly by modifying the expirationTime attribute and that it will not be implicitly extended by accessing the <*container*> or any of its descendant resources.

Issuer: shall request to update the <*container*>resource, using the UPDATE verb. The request shall address a <*container*> resource as defined in clause 9.2.3.12. The issuer may send new (proposed) values for all mandatory read-write attributes and may send values for the optional read-write attributes. They will be handled similarly as in the create request.

The issuer can be an application or an SCL.

Hosting SCL: shall validate the received request. The update shall be allowed if the issuer is authorized to modify the <*container*> resource, according to the accessRight defined for the <*container*> resource where the request is targeted. The hosting SCL shall then modify the resource representation according to the received attributes. Like in the Create, the hosting SCL may modify some of the attributes, e.g. due to internal policies. The hosting SCL shall provide default values for not provided mandatory attributes. The update of the <*container*> resource may result in announcement and de-announcements, depending on the new value of the announceTo attribute, see clause 9.2.3.12 for details.

If the maxNrOfInstances, maxByteSize or maxInstanceAge are changed in the <*container*> resource, the hosting SCL shall immediately enforce the new restrictions. This may potentially lead to the deletion of the oldest <*contentInstance*> resources.

The hosting SCL shall return a generic response according to clause 9.3.1.1.

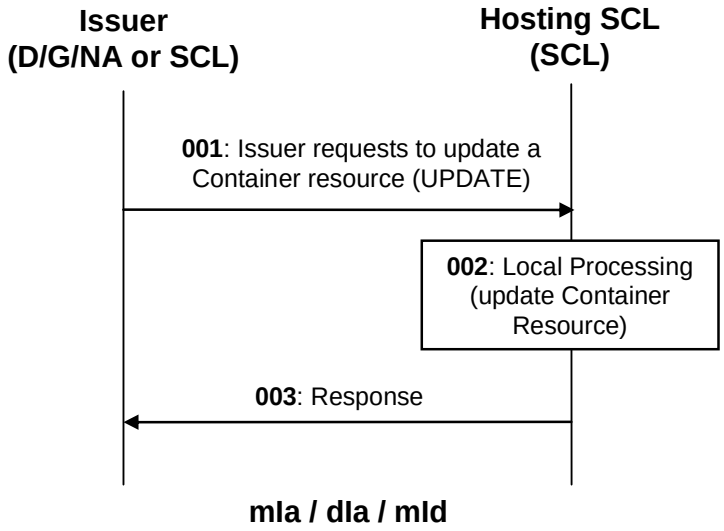


Figure 9.65: Message flow of procedure to update the complete <*container*> resource

Step 001: To update a <*container*>resource, the issuer shall issue an UPDATE request, providing the address of the <*container*>resource.

Step 002: If the update operation is allowed by Hosting SCL, the requested <*container*>resource shall be updated.

The issuer shall include values for all Mandatory RW attributes, and optionally include values for Optional RW attributes.

Step 003: A result shall be returned as a response to the issuer.

List of main procedure specific exceptions:

Step 002: The requesting application or SCL is not registered.

Step 002: The addressed resource does not exists.

Step 002: The requesting application or SCL does not have the authorization to update this resource.

Step 002: The issuer did not include values for all mandatory RW attributes.

Step 002: The values provided in the request are not valid.

Step 002: The attributes in the request are not RW so they cannot be updated.

9.3.2.12.5 Delete <container>

This procedure shall be used to delete a specific <*container*> resource.

Issuer: requests to delete a <*container*>resource using the DELETE verb. The request shall address a <*container*>resource of an SCL as defined in clause 9.2.3.12.

The issuer can be an application or an SCL.

Hosting SCL: shall validate the received request. The Delete shall be allowed if the issuer is authorized to delete the <*container*>resource, according to the accessRight defined for the <*container*>resource where the request is targeted. The hosting SCL then shall delete the addressed <*container*>resource. The hosting SCL shall return a generic response according to clause 9.3.1.1.

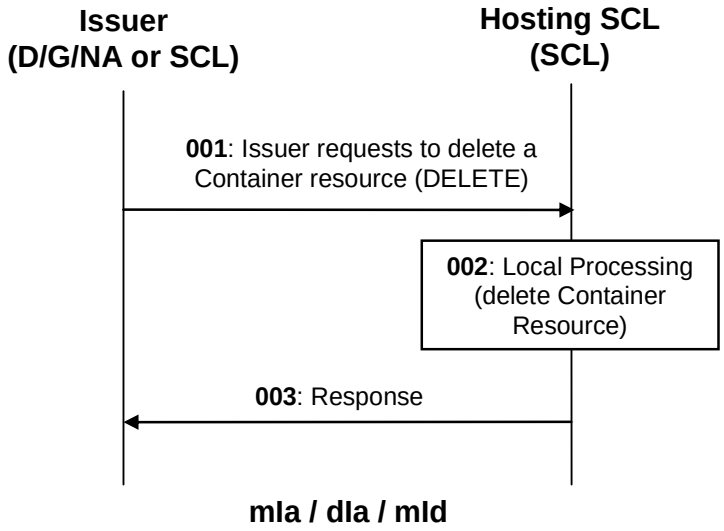


Figure 9.66: Message flow of procedure to delete a <*container*> resource

Step 001: To delete a <*container*>resource, the issuer shall issue a DELETE request, providing the address of the <*container*>resource to be deleted.

Step 002: If the delete operation is allowed by Hosting SCL, the addressed <*container*>resource shall be deleted.

Step 003: A result shall be returned as a response to the issuer.

List of main procedure specific exceptions:

Step 002: The requesting application or SCL is not registered.

Step 002: The addressed resource does not exists.

Step 002: The requesting application or SCL does not have the authorization to delete this resource.

9.3.2.12.6 Subscribe/Unsubscribe to <*container*>

This procedure shall be used to (un)subscribe to modification in the attributes of a <*container*>.

The procedure is described in details in clause 9.3.2.18.

9.3.2.12.7 Create <*containerAnnc*> (Announce/De-announce a <container>)

This procedure shall be used to (de-)announce a <*container*> resource to a remote SCL.

These procedures manipulate (create/delete) a resource <*containerAnnc*> in the Announce-To SCL.

The procedure is described in details in clause 9.3.2.28.

9.3.2.13 Location Container management

9.3.2.13.1 Introduction

This clause describes different procedures for managing the creation, retrieval, updating and deletion of information associated with a <*locationContainer*> resource. Since a <*locationContainer*>resource is very similar to a <*container*> resource, the procedures in this clause only highlight the differences with the corresponding procedures for the <*container*> resource.

9.3.2.13.2 Create <locationContainer>

This procedure is used for creating a <*locationContainer*> resource.

A <*locationContainer*> resource can only be created under a *containers* resource of which the parent resource is <*application*> or <*applicationAnnc*> resource. The place where a <*locationContainer*> resource can be created also depends on the *locationContainerType* attribute in the create request. The following rules shall apply.

If the *locationContainerType* attribute in the create request indicates "application generated" then resource shall be allowed to be created at the following locations:

<d/gsclBase>/applications/<app>/containers or
<nsclBase>/applications/<app>/containers or
<d/gsclBase>/scls/<scl>/applications/<applicationAnnc>/containers
<nsclBase>/scls/<scl>/applications/<applicationAnnc>/containers

If the *locationContainerType* attribute in the create request indicates "location server based" then resource shall be allowed to be created at the following location:

<nsclBase>/scls/<scl>/applications/<applicationAnnc>/containers

Issuer: shall request to create a new <*locationContainer*>resource using a CREATE verb. The request is addressing a URI of a *containers* resource as described in clause 9.2.3.11, limited to the locations described above.

The request shall also provide a *locationContainterType* attribute. This attribute indicates the source of the location information, i.e. whether it is application generated or location server based.

The issuer can be an application or an SCL.

Hosting SCL: shall follow the procedures as described in clause 9.3.2.12.2, except that issuer requests to create a <*locationContainer*> resource.

9.3.2.13.3 Retrieve <locationContainer>

This procedure is used for retrieving a <*locationContainer*> resource representation. It is performed as retrieving a <*container*> resource as described in clause 9.3.2.12.3, except that the addressed resource is a <*locationContainer*> resource.

9.3.2.13.4 Update <locationContainer>

This procedure is used to modify the attributes defined in a <*locationContainer*> resource.

It is performed as updating a <*container*> resource as described in clause 9.3.2.12.4 except that the addressed resource is a <*locationContainer*> resource.

9.3.2.13.5 Delete <locationContainer>

This procedure is used to delete a <*locationContainer*> resource. It is performed as deleting a <*container*> resource as described in clause 9.3.2.12.5 except that the addressed resource is a <*locationContainer*> resource.

9.3.2.13.6 Subscribe/Un-subscribe to <*locationContainer*>

This procedure shall be used to (un)subscribe to modification in the attributes of a <*locationContainer*> resource.

The procedure is described in details in clause 9.3.2.18.

9.3.2.13.7 Create <*locationContainerAnnc*> (Announce/De-announce a <locationContainer>*)*

This procedure shall be used to (de-)announce a <*locationContainer*> resource to a remote SCL.

These procedures manipulate (create/delete) a resource <*locationContainerAnnc*> in the Announce-To SCL.

The procedure is described in details in clause 9.3.2.28.

9.3.2.14 Content Instances collection management

9.3.2.14.1 Introduction

The *contentInstances* resource is a sub-resource of both the <*container*> and the <*locationContainer*> resources. It gives access to the <*contentInstance*> resources and also maintains some dynamic information about the total content in the <*container*>, such as the current number of <*contentInstance*> resources and the total size of all the content data of all the current <*contentInstance*> resources combined.

The *contentInstance* resource is special, in that this is the only collection resource, whose representation includes the (full or partial) representations of its child resources, whereas the other collection resources only represent their children by their URIs.

This clause shows how to retrieve and update the *contentInstances* collection resource. The procedures depend on whether the parent resource is a <*container*> or a <*locationContainer*> instance and on the *locationContainerType* attribute of the parent <*locationContainer*> resource.

9.3.2.14.2 Retrieve instances in *contentInstances*

This procedure shall be used to get the information of *contentInstances* resource together with the data of all <*contentInstance*> resources in the addressed *contentInstances* collection resource that match the specified filter criteria.

The precondition for this procedure is that the metadata-only parameter in the retrieve request is set to FALSE.

The result includes both the meta-data, the actual (opaque) content included in the <*contentInstance*> child-resource that match the filter criteria.

Meta-data includes the content creation time, the size, an optional content-type and the URI of the <*contentInstance*> resource that can be used to directly access that specific contentInstance.

Issuer: shall request to retrieve the information of a *contentInstances* collection resource child of the <*container*>, using the RETRIEVE verb. The request shall address a *contentInstances* collection resource of an SCL as defined in clause 9.2.3.16. The request may contain filter criteria. If no filter criteria are present, then the data of all <*contentInstance*> resources in the collection are requested. A Filter criteria can for example apply to the creation time of the instances (i.e. return only <*contentInstance*> resources creates after a certain time and date).

The issuer can be an application or an SCL.

Hosting SCL: shall validate the received request. Retrieval shall be allowed if the issuer is authorized to retrieve the *contentInstances* resource, according to the accessRight defined for the parent <*container*> resource of the *contentInstance* collection where the request is targeted. The hosting SCL shall return a representation of the *contentInstances* resource, with all the attributes defined in the collection resource and a list representing the data of all <*contentInstance*> child resources that match the indicated filter criteria. If no filter criteria are provided, data of all the <*contentInstance*> child resources shall be returned. The hosting SCL shall return a generic response as indicated in clause 9.3.1.1.

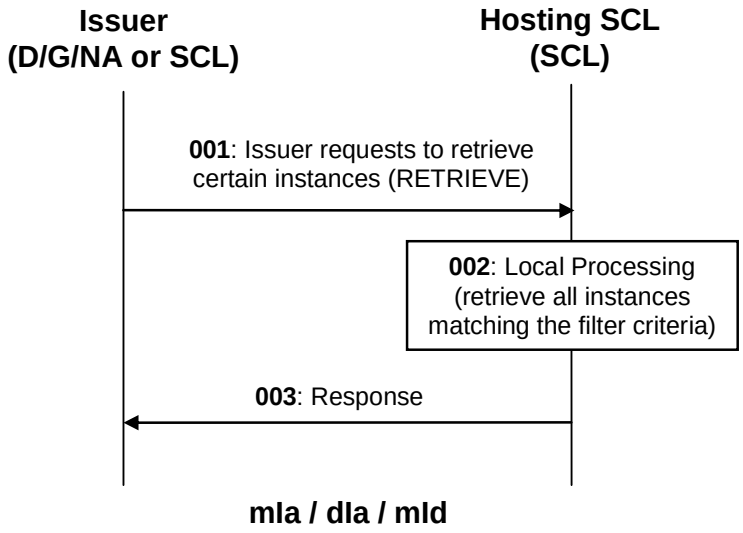


Figure 9.67: Message flow of procedure to read all instances matching filter criteria in a <*container*> resource

Step 001: To read *contentInstances*, the issuer shall issue a RETRIEVE request. The issuer may include filter criteria.

Step 002: If the retrieve operation is allowed by Hosting SCL, the requested *contentInstances* shall be retrieved.

Step 003: A result shall be returned as a response to the issuer, containing the retrieved information.

List of main procedure specific exceptions:

Step 002: The requesting application or SCL is not registered.

Step 002: The addressed resource does not exists.

Step 002: The requesting application or SCL does not have the authorization to read this resource.

9.3.2.14.3 Retrieve meta-data from *contentInstances*

This procedure shall be used to get a list of meta-data of all instances in the addressed *contentInstances* collection resource, that match specified filter criteria. The idea is that the meta-data gives enough information for the issuer to retrieve one or more specific <*contentInstance*> resources.

The precondition for this procedure is that the metadata-only attribute in the request is set to TRUE.

Issuer: shall request to retrieve the meta-information of a *contentInstances* collection resource using a RETRIEVE verb. The request shall address a *contentInstances* collection resource of an SCL as defined in clause 9.2.3.16. The request indicates that only meta-data of the contained <*contentInstance*> resources shall be returned. The request can optionally contain filter criteria. If no filter criteria are present, then the meta-data of all <*contentInstance*> resources in the collection will be returned. Filter criteria can, for example, apply to the creation time of the instances, the size of the instances or the content-type of the instances.

The issuer can be an application or an SCL.

Hosting SCL: validates the received request. Retrieval shall be allowed if the issuer is authorized to retrieve the *contentInstances* resource, according to the accessRight defined for the parent <*container*> resource of the addressed collection. The hosting SCL shall return a representation of the *contentInstances* resource, with all the attributes defined in the collection resource and a list representing the resource representation of all the child <*contentInstance*> resources that match the indicated filter criteria. The resource representation includes the meta-data about those <*contentInstance*> resources. The hosting SCL shall *not* include the actual content data included in the <*contentInstance*> resources (i.e. it will exclude the *content* attribute in each <*contentInstance*> resource representation). If no filter criteria are provided, meta-data of all the children are returned.

The hosting SCL shall return a generic response as indicated in clause 9.3.1.1.

No specific flow is provided for this case. See the call flow in "Read all instances in a <*container*> matching certain filter criteria".

9.3.2.14.4 Retrieve instances from *contentInstances* of <*locationContainer*> of type "application generated" matching filter criteria

This procedure applies when a *contentInstances* resource is addressed whose <*locationContainer*> parent resource has a *locationContainerType* attribute that indicates "application generated" and the metadata-only attribute is set to FALSE.

The procedure is the same as described in "retrieve instances in a <*container*> resource matching filter criteria", with the understanding that the <*contentInstance*> resources content data should contain location data related to the application resource under which the <*locationContainer*> resource resides.

9.3.2.14.5 Retrieve meta-data of instances in a <*locationContainer*> of type "application generated" matching filter criteria

This procedure applies when a *contentInstances* resource is addressed whose <*locationContainer*> parent has a *locationContainerType* attribute that indicates "application generated" and the metadata-only attribute is set to TRUE.

The procedure is the same as described in "retrieve meta-data of instances in a <*container*> resource matching filter criteria", with the understanding that the <*contentInstance*> resources meta-data is about location data related to the application resource under which the <*locationContainer*> resides.

9.3.2.14.6 Retrieve instances from *contentInstances* of <*locationContainer*> of type "location server based"

This procedure applies when a *contentInstances* resource is addressed whose <*locationContainer*> parent has a *locationContainerType* attribute that indicates "location server based".

This procedure is used to retrieve information of the *contentInstances* resource together with the server generated location data of the application. In this procedure the filter criteria are not applicable and any request containing them will be rejected.

Issuer: shall request to retrieve the information of a *contentInstances* collection resource of a <*locationContainer*> resource, using the RETRIEVE verb as described in clause 9.3.2.13.3.

The issuer shall only be an Application.

Hosting SCL: shall validate the received request. Retrieval shall be allowed if the issuer is authorized to retrieve the *contentInstances* resource, according to the accessRight defined for the parent <*locationContainer*> resource of the collection where the request is targeted.

If any filter criteria are present the hosting SCL shall reject the request.

The hosting SCL shall determine the source of location information by the *locationContainerType* attribute of the parent <*locationContainer*> resource. If the *locationContainerType* attribute indicates the location information is provided by a location server (e.g. 3GPP location server), the hosting SCL shall first retrieve the *locTargetDevice* attribute from the <*scl*> resource under which the <*locationContainer*> resource is located to obtain the address of the target device which is represented by this <*scl*> resource.

Furthermore, the hosting SCL shall also retrieve the *locRequestor* attribute from the issuer (i.e. an <*application*> resource) to obtain the identity of the <*application*> to be used for the consent of location privacy control.

In case either the *locTargetDevice* or the *locRequestor* cannot be obtained, the hosting SCL shall reject the request.

Then the hosting SCL shall transform the RESTful request received from the issuer into LCS Service (LoCation Services) request including the retrieved *locTargetDevice* and *locRequestor* information. The interface with the location server is out of scope of the present document. For example, it could be the OMA Mobile Location Protocol [11].

The hosting SCL shall also provide default values for other parameters (e.g. location accuracy) required in the LCS service request [12] according to local policies. The location server performs the privacy control and if permitted positioning procedures [12], and returns a response to the hosting SCL. The hosting SCL receives the corresponding LCS service response and transforms it into RESTful response and return to the issuer with the location information only in case of success response. The location data retrieved from the location server is reported in the response as the content data of a not addressable *contentInstance* resource.

The hosting SCL shall return a generic response as indicated in clause 9.3.1.1.

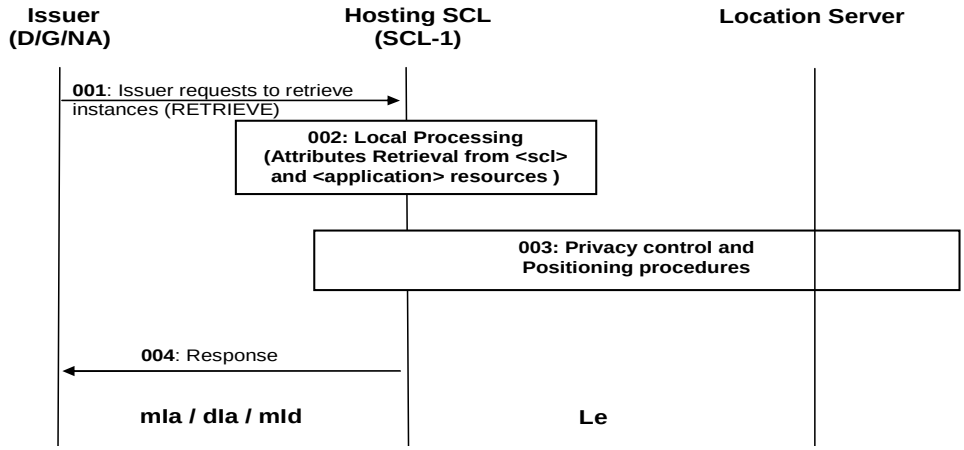


Figure 9.68: Illustration of Procedures for the case of location server based to read all instances in a <*locationContainer*> resource

Step 001: To read *contentInstances*, the issuer shall send a RETRIEVE request. The request shall include the parameters as above.

Step 002: If the operation is allowed by hosting SCL. The hosting SCL shall first retrieve the *locTargetDevice* attribute of the <*scl*> which represents the target device to be located and the *locRequestor* of the issuer <*application*> as described above.

Step 003: The hosting SCL shall then interact with location server through the interface out of scope, e.g. Le interface [12]. Then the location server shall respond to the hosting SCL with a response [11], [12].

Step 004: The hosting SCL constructs a *contentInstances* resource representation based on the received information and shall respond to the issuer with the appropriate generic responses. The location information may be returned in an asynchronous manner, referring to the semi-asynchronous mechanism as described in clause 9.3.1.4.

List of procedure specific exceptions:

Step 002: The requesting application is not registered.

Step 002: The issuer of the request is not an application.

Step 002: The request is not valid (e.g. the metadata-only parameter is present).

Step 002: The addressed resource does not exists.

Step 002: The requesting application or SCL does not have the authorization to read this resource.

Step 003: Issuer is not allowed to locate the M2M Device/Gateway according to the result of privacy control performed in step 003.

Step 002: Optional attributes needed for location are not available.

Step 003: The request to the LCS is unsuccessful.

Illustration of procedures for the case of M2M application generated is described in clause 9.3.2.13.3.

9.3.2.14.7 Retrieve meta-data from <*contentInstances*> of <*locationContainer*> matching of type "server generated" matching filter criteria

This procedure applies when a *contentInstances* resource is addressed whose <*locationContainer*> parent has a *locationContainerType* attribute that indicates "location server based" and where the meta-data only attribute is set to TRUE.

The hosting SCL will reject such a request, since a location server based location is dynamic and therefore requesting only the meta-data is not useful.

9.3.2.14.8 Subscribe/Unsubscribe to a *contentInstances* in a <*container*>

This procedure shall be used to (un)subscribe to modification to the *contentInstances*, i.e. the addition of new <*contentInstance*> resource or the removal of <*contentInstance*> resources.

The procedure is described in details in clause 9.3.2.18.

9.3.2.14.9 Subscribe/Unsubscribe to contentInstances in a <locationContainer>

This procedure shall be used to (un)subscribe to modification to the *contentInstances* of a parent <*locationContainer*> resource, i.e. the addition of new <*contentInstances*> resources or the removal of <*contentInstance*> resources. It shall be performed as subscribing/unsubscribing to *contentInstances* of a <*container*> resource as described in clause 9.3.2.12.6, except that the addressed resource is the *contentInstances* resource of a <*locationContainer*> resource.

In the case the *locationContainerType* attribute of the parent <*locationContainer*> resource indicated location server based location, when some entity retrieves the *contentInstances* or retrieves a specific <*contentInstance*> of the parent <*locationContainer*> resource as described in clauses 9.3.2.14.6 and 9.3.2.14.9, the subscriber shall be notified if the hosting SCL receives the location information in success response from the location server.

9.3.2.15 Content Instance management

9.3.2.15.1 Introduction

The <*contentInstance*> resource is a resource that embodies the actual opaque data that is exchanged via a <*container*> or <*locationContainer*> resource. It also contains meta-data associated with this data. This can be used to gauge the interest in the data, before getting the actual content data, since the size of the contentInstance's opaque data can be quite large.

A <*contentInstance*> resource can be addressed individually, but its representation is also a part of the parent *contentInstances* resource representation. This allows the retrieval of multiple <*contentInstance*> resources with one REST request.

This clause shows how to create, retrieve and delete the <*contentInstance*> resource. Since <*contentInstance*> resources are immutable, there is no procedure for updating.

Some of the procedures depend on whether the <*contentInstance*> resource located under a <*container*> resource, a <*locationContainer*> resource with a *locationContainerType* of "application generated" or a <*locationContainer*> resource with a *locationContainerType* of "location server based".

9.3.2.15.2 Create <contentInstance> in a <container>

This procedure shall be used to add a <*contentInstance*> resource to a <*container*> resource. It is modelled as creating an <*contentInstance*> resource in the *contentInstances* collection resource of that <*container*> resource. The information contained in the <*contentInstance*> shall be opaque to the hosting SCL. The information may even be encrypted.

Issuer: requests to create a <*contentInstance*> resource using the CREATE verb. The request shall address a *contentInstances* collection resource in a <*container*> resource as defined in clause 9.2.3.12.

The issuer may include the following information in the request; the actual (opaque) content, a delayTolerance and some meta-data, like content-type.

The issuer can be an application or an SCL.

Hosting SCL: shall validate the received request and shall create a <*contentInstance*> resource with the specified attributes. Creation shall be allowed if the issuer is authorized to create a child resource according to the accessRight defined for the parent <*container*> resource of the *contentInstances* collection resource where the request is targeted. The hosting SCL shall check the policy restrictions as specified on the parent <*container*> resource. If the newly added instance violates any of the policies (e.g. the total size becomes too large), then the oldest instance resources shall be removed from the collection until the policies are satisfied. Then the hosting SCL shall change the link-resources "latest" and "oldest" to refer to the newly added resource and the oldest resource respectively.

After creation of the <*contentInstance*> resource, the hosting SCL shall return a generic response as indicated in clause 9.3.1.1. The response shall include the URI of the created <*contentInstance*> resource.

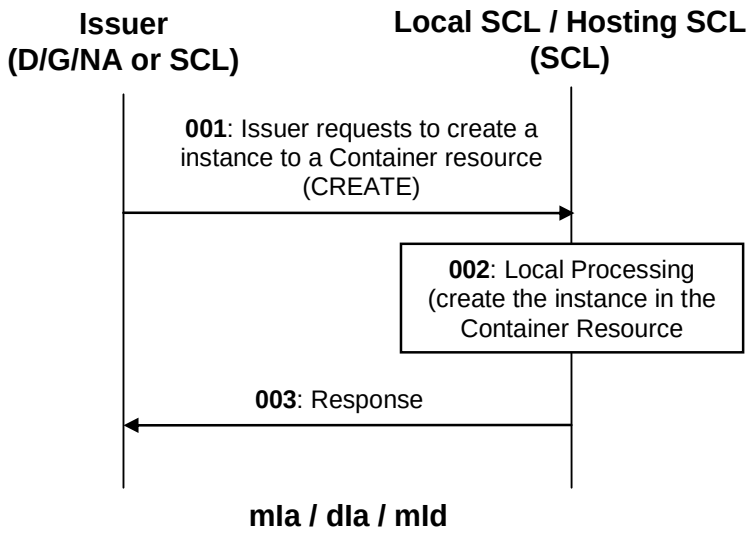


Figure 9.69: Message flow of procedure to add an instance to a <*container*> resource

Step 001: To add an <*contentInstance*> resource to a <*container*> resource, the issuer shall issue a CREATE request. The request addresses the *contentInstances* child resource of the <*container*>.

Step 002: If the creation is allowed by Hosting SCL, the <*contentInstance*> resource under the *contentInstances* resource residing under the <*container*> resource shall be created.

Step 003: A result shall be returned as a response to the issuer.

List of main procedure specific exceptions:

Step 002: The requesting application or SCL is not registered.

Step 002: The requesting application or SCL does not have the authorization to create this resource.

Step 002: The provided characteristics are not acceptable to the Hosting SCL.

Step 002: The <*container*> resource does not exist.

9.3.2.15.3 Create <contentInstance> in a <locationContainer>

This procedure is used to add an <*contentInstance*> resource to a <*locationContainer*> resource that contains location information generated from an M2M application. It is performed exactly as adding an instance to a <*container*> resource described in detail in clause 9.3.2.15.2, except that the addressed resource is the *contentInstances* child resource of a <*locationContainer*> resource. Furthermore, if issuer sends a request to add an instance to a <*locationContainer*> resource that contains a *locationContainerType* attribute indicating the case of location server based, the request shall be rejected.

9.3.2.15.4 Retrieve <contentInstance> from a <*container*>

This procedure shall be used to retrieve the attributes/meta-data and the opaque data of a <*contentInstance*> resource residing under a <*container*> resource.

Issuer: shall request to retrieve the information of a <*contentInstance*> resource, using the RETRIEVE verb. The request shall address an existing <*contentInstance*> resource of an SCL as defined in clause 9.2.3.17.

The issuer can be an application or an SCL.

Hosting SCL: validates the received request. Retrieval shall be allowed if the issuer is authorized to retrieve the <*contentInstance*> resource, according to the accessRight defined for the <*container*> resource that is the grand-parent of the instance where the request is targeted. The hosting SCL shall return a representation of the <*contentInstance*> resource, with all the attributes defined in the instance, including the content.

See the call flow in "Read all instances in a <*container*> matching certain filter criteria".

9.3.2.15.5 Retrieve <contentInstance> from a <*locationContainer*>

This procedure is used to retrieve the attributes and the opaque data of a <*contentInstance*> resource of a <*locationContainer*> resource.

Issuer: requests to retrieve the information of an <*contentInstance*> resource of a <*locationContainer*> resource of an SCL as described in clause 9.2.3.17.

The issuer can be an application or an SCL.

Hosting SCL: shall activate the validation for the received request. Retrieval shall be allowed if the issuer is authorized to retrieve the <*contentInstance*> resource, according to the accessRight defined for the grand-parent <*locationContainer*> resource of the instance where the request is targeted. The hosting SCL shall determine the source of location information by the *locationContainerType* attribute of the grand-parent <*locationContainer*> resource. If the *locationContainerType* attribute indicates the location information is generated from an application, the hosting SCL shall retrieve and return the location related contentInstance as described in clause 9.2.3.17; if the *locationContainerType* attribute indicates the location information is from location server, e.g. 3GPP location server, and the addressed URI is the latest or oldest link (i.e. …./<*locationContainer*>*/contentInstances/latest* or …./<*locationContainer*>*/contentInstances/oldest*) the hosting SCL shall retrieve and return the location instance as described in clause 9.3.2.13.3, with the exception that the returned non-addressable <*contentInstance*> resource representation is not embedded in a *contentInstances* resource representation.

9.3.2.15.6 Delete <contentInstance>

This procedure shall be used to delete an <*contentInstance*> resource.

Issuer: requests to delete an <*contentInstance*> resource, using the DELETE verb. The request shall address an <*contentInstance*> resource of an SCL as defined in clause 9.2.3.17.

Hosting SCL: validates the received request. Deletion shall be allowed if the issuer is authorized to delete the <*contentInstance*> resource, according to the accessRight defined for the <*container*> resource grand-parent of the instance where the request is targeted. The hosting SCL shall delete the resource and if needed updates the latest and oldest links. It shall return a generic response as described in clause 9.3.1.1.

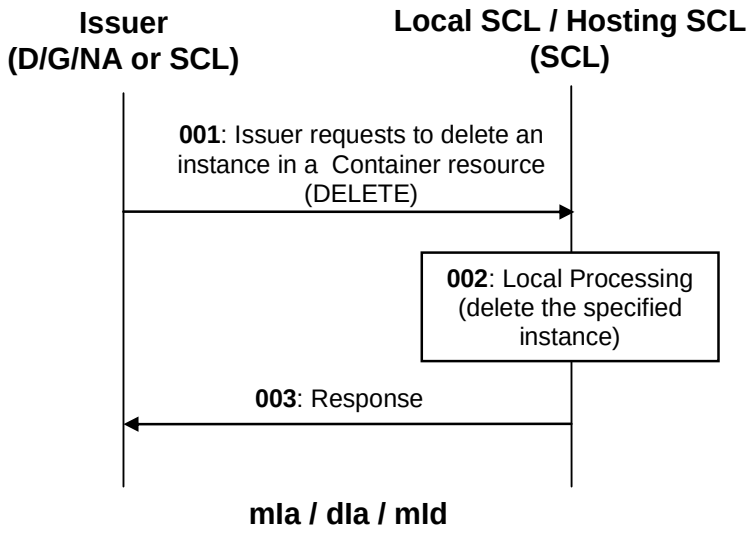


Figure 9.70: Message flow of procedure to delete an instance in a <*container*> resource

Step 001: To delete a <*contentInstance*> in a <*container*> resource, the issuer shall issue a DELETE request, providing the address of the <*contentInstance*> to be deleted.

Step 002: If the delete operation is allowed by Hosting SCL, the requested <*contentInstance*> resource shall be deleted.

Step 003: A result shall be returned as a response to the issuer.

List of main procedure specific exceptions:

Step 002: The requesting application or SCL is not registered.

Step 002: The addressed contentInstance resource does not exists.

Step 002: The requesting application or SCL does not have the authorization to delete this resource.

9.3.2.16 Group collection management

9.3.2.16.1 Introduction

This clause describes different procedures for managing the retrieval and updates of information associated with a *groups* resource as defined in clause 9.2.3.18. Such type of resource (like any other collection) cannot be created or deleted by means of a request over the reference points: mIa, mId and dIa.

9.3.2.16.2 Retrieve *groups*

This procedure shall be used for getting all attributes of a *groups* collection resource and the references to visible child resources.

The procedure is described in details in clause 9.3.2.30.2.

9.3.2.16.3 Update *groups*

This procedure shall be used for modifying the attributes in a *groups* collection resource.

The procedure is described in details in clause 9.3.2.30.3.

9.3.2.16.4 Subscribe/Un-Subscribe *groups*

These procedures shall be used for subscribing and un-subscribing for changes in a *groups* collection resource.

The procedures are described in details in clause 9.3.2.30.6.

9.3.2.17 Group management

9.3.2.17.1 Introduction

This clause describes different procedures for managing membership verification, creation, retrieval, update, deletion, subscription and announcement of the information associated with an <*group*> resource as well as the bulk management of all group member resources by invoking the corresponding verbs upon the virtual sub-resource *membersContent* of a <*group*> resource.

9.3.2.17.2 Create <*group*>

This procedure shall be used for creating a group resource with one or multiple members or without members.

Issuer: shall request to create a new group type resource to be named as <*group*> by using the CREATE verb. The request shall address *groups* resource of an hosting SCL. The request shall also provide list of member URI and may provide *expirationTime* and *searchStrings* attributes. The list of member URI means a list of URIs of the *member* resources corresponding to the *memberType* attribute provided in the request. The issuer may be an application or an SCL.

Hosting SCL: shall check if the issuer has CREATE permissions on the *groups* resource. The hosting SCL shall also check the validity of provided attributes and verify that none of the provided member URI refer to a remote <*group*> resource (hosted by a remote SCL). The hosting SCL shall also validate that the resource type of any member conforms to the *memberType* attribute of the <*group*> resource, if the *memberType* attribute of the <group> resource is not 'mixed'. Upon successful validation, a new group resource with name <*group*> including the "*membersContent"* sub-resource shall be created in the hosting SCL. Then the SCL shall respond to the issuer with the appropriate generic responses described in clause 9.3.1.1. The SCL shall also provide in the success response the URI of the created <*group*> resource. If the *memberType* validation fails, the hosting SCL shall deal with the <group> resource according to the policy defined by the *consistencyStrategy* attribute of the <group> resource provided in the request.

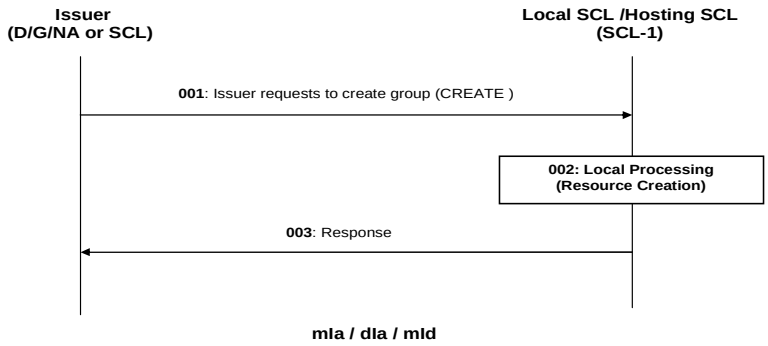


Figure 9.71: Illustration of Procedures to Create Group

Step 001: To create a group resource, the issuer shall send a create group request to an SCL using the CREATE verb.

Step 002: If the operation is allowed by Hosting SCL the <*group*> resource shall be created. If no characteristics are provided by the issuer, the <*group*> resource shall be created with default characteristics determined by the Hosting SCL.

Step 003: Then the SCL shall respond to the issuer with the appropriate generic responses.

List of main procedure specific exceptions:

Step 002: Issuer does not have the authorization to create this resource.

Step 002: The provided characteristics are not acceptable to Hosting SCL.

9.3.2.17.3 Retrieve <*group*>

This procedure shall be used for retrieving <*group*> resource. Alternatively, the issuer can request to retrieve only a specific attribute or part of an attribute, as described in partial addressing, see clause 9.3.2.29.

Issuer: shall request to obtaining <*group*> resource information by using the RETRIEVE verb. The request shall address the specific <*group*> resource of an hosting SCL. The issuer may be an application or an SCL.

Hosting SCL: shall check if the issuer has READ permission on the group resource. Upon successful validation, the hosting SCL shall respond to the issuer with the appropriate responses described in clause 9.3.1.1 and the resource representation.

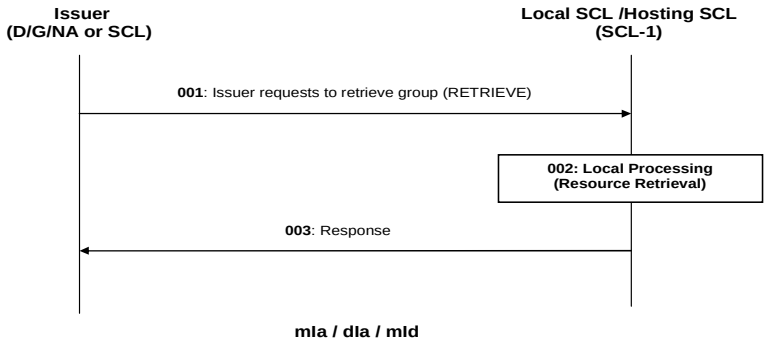


Figure 9.72: Illustration of Procedures to Retrieve Group

Step 001: To retrieve a group resource, the issuer shall send a retrieve group request to an SCL using a RETRIEVE verb.

The request shall include the parameter as above. The request shall also include authentication and/or authorization data.

Step 002: If the operation is allowed by Hosting SCL the group resource shall be retrieved.

Step 003: Then the SCL shall respond to the issuer with the appropriate generic responses.

List of main procedure specific exceptions:

Step 002: Issuer is not registered.

Step 002: Issuer does not have the authorization to retrieve this resource.

Step 002: Group resource does not exist.

9.3.2.17.4 Update <*group*>

This procedure shall be used for updating an existing <*group*> resource. Alternatively, the issuer can request to update only a specific attribute or part of an attribute, as described in partial addressing, see clause 9.3.2.29.

Issuer: shall request to update attributes of an existing <*group*> resource by using an UPDATE verb. The request shall address the specific <*group*> resource of an SCL. The issuer may send new (proposed) values for all mandatory read-write attributes and may send values for the optional read-write attributes. They will be handled similarly as in the create request. The issuer may be an application or an SCL.

Hosting SCL: shall check if the issuer has WRITE permission on the <*group*> resource. The hosting SCL shall also check the validity of provided attributes and verify that none of the provided member URI refer to a remote <*group*> resource (hosted by a remote SCL). The hosting SCL shall detect the member and descendent members of the local <*group*> resource to ensure that the existing <*group*> resource is not contained in the local <*group*> resource. The descendent members means the members of sub-group resources if the local <*group*> resource contains sub-group resources which may also contains multi-level sub-group resources. The hosting SCL shall also validate that the resource type of any member conforms to the *memberType* attribute of the <*group*> resource, if the *memberType* attribute of the <group> resource is not 'mixed'. Upon successful validation, the hosting SCL shall modify the attributes accordingly and shall respond to the issuer with the appropriate generic responses described in clause 9.3.1.1. If the *memberType* validation fails, the hosting SCL shall deal with the <group> resource according to the policy defined by the *consistencyStrategy* attribute of the <group> resource provided in the request.

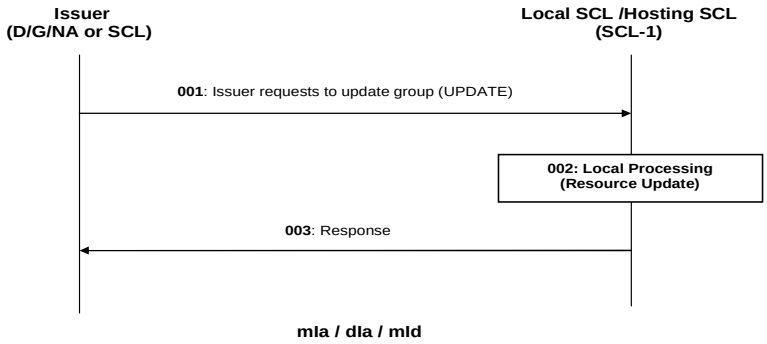


Figure 9.73: Illustration of Procedures to Update Group

Step 001: To update a group resource, the issuer shall send a update group request to an hosting SCL using the UPDATE verb for full update or UPDATE, CREATE or DELETE in case of partial update.

Step 002: If the operation is allowed by Hosting SCL the group resource shall be updated.

Step 003: Then the SCL shall respond to the issuer with the appropriate generic responses.

List of main procedure specific exceptions:

Step 002: Issuer does not have the authorization to update this resource.

Step 002: Group resource does not exist.

9.3.2.17.5 Delete <*group*>

This procedure shall be used for deleting a existing <*group*> resource.

Issuer: shall request to delete an existing <*group*> type resource by using the DELETE verb. The request shall address the specific <*group*> resource of an hosting SCL. The issuer may be an application or an SCL.

Hosting SCL: shall check if the issuer has DELETE permission on the <*group*> resource. Upon successful validation, the SCL shall remove the resource from its repository and shall respond to the issuer with the appropriate responses described in clause 9.3.1.1.

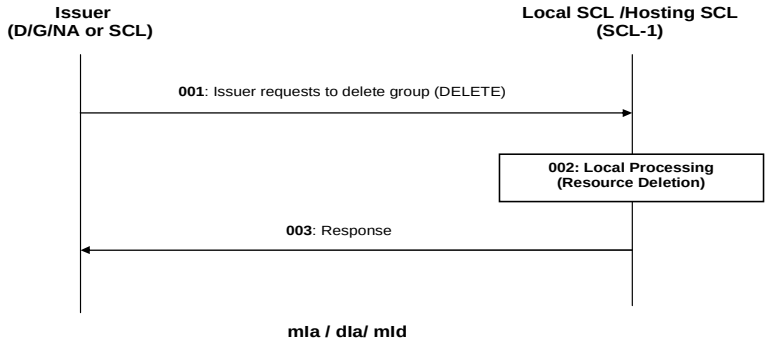


Figure 9.74: Illustration of Procedures to Delete Group

Step 001: To delete a group resource, the issuer shall send a delete group request to an SCL using a DELETE verb.

Step 002: If the operation is allowed by Hosting SCL the group resource shall be deleted from its repository.

Step 003: Then the SCL shall respond to the issuer with the appropriate generic responses.

List of main procedure specific exceptions:

Step 002: Issuer is not registered.

Step 002: Issuer does not have the authorization to delete this resource.

Step 002: Group resource does not exist.

9.3.2.17.6 Subscribe/Un-subscribe to <*group*>

These procedures shall be used for subscribing for changes in a <*group*> resource and managing the subscription itself. For the subscription to the content of all member resources of a <*group*>, see clause 9.2.3.19.

The procedures are described in details in clause 9.3.2.19.

9.3.2.17.7 Create <*groupAnnc*> (Announce/de-announce a <*group*>)

These procedures shall be used for announcing and/or de-announcing a <*group*> resource to/from another SCL.

The procedure is described in details in clause 9.3.2.28.

9.3.2.17.8 Verify group membership

This procedure shall be used for verifying that a specific resource is a member of an indicated group of resources, i.e. to confirm a membership of a resource.

Issuer: shall request to verify whether a given resource with the URI of <*memberId*> belongs to a specific <*group*> resource by using the RETRIEVE verb. The request shall address the specific <*memberId*> attribute of the <*group*> resource of a hosting SCL. The issuer may be an application or an SCL.

Hosting SCL: shall validate if the issuer has READ permission on the <*group*> resource. Upon successful validation the hosting SCL shall verify that the given resource is one of the <*memberId*> belonging to the *members* attribute of the <*group*> resource. Then the SCL shall respond to the application with the appropriate response described in clause 9.3.1.1 and indicate whether the <*memberId*> was present in the *members* collection attribute or not.

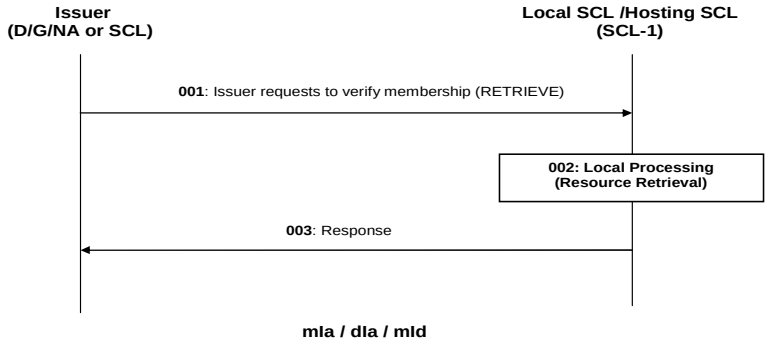


Figure 9.75: Illustration of Procedures to Verify Membership

Step 001: To verify that a specific resource is a member to an indicated group of resources, the issuer shall send a verify membership request to an SCL using a RETRIEVE verb.

The request shall include the parameters as mentioned above.

Step 002: If the operation is allowed by hosting SCL the group resource shall be retrieved as described above.

Step 003: Then the SCL shall respond to the issuer with the appropriate generic responses.

List of main procedure specific exceptions:

Step 002: Issuer does not have the authorization to retrieve the group resource.

Step 002: ResourceURI does not exist as a memberId in the indicated group resource.

Step 002: The group resource does not exist.

9.3.2.17.9 Add/Delete a specific member to/from a group

This procedure shall be used for adding a new member into an existing group resource or deleting a member from an existing group resource. This procedure is supported by means of partial addressing, see clause 9.3.2.29.

9.3.2.17.10 Retrieve all members

This procedure shall be used for retrieving whole member list of an existing group resource. This procedure is supported by means of partial addressing, see clause 9.3.2.29.

9.3.2.17.11 Delete all members

This procedure shall be used for remove all members from an existing group resource. The attribute *members* in the existing group resource is not deleted, but only reset to an empty list. This procedure is supported by means of partial addressing, see clause 9.3.2.29.

9.3.2.17.12 Create membersContent

This procedure shall be used for creating the content of all member resources belonging to an existing <*group*> resource.

Issuer: shall request to create the content in all member resources belonging to an existing <*group*> resource by using a CREATE verb. The request may address the virtual sub-resource *membersContent* of the specific <*group*> resource of a group hosting SCL to create the same content under all member resources. The request may also address the URI that results from appending a relative URI to the *membersContent* URI in order to create the same content (e.g. attribute or sub-resource) under the corresponding attributes or sub-resources represented by the relative URI with respect to all member resources. The issuer may be an application or SCL.

Group Hosting SCL: shall check if the issuer has WRITE permission in the accessRight resource referenced by the membersContentAccessRightID on the group resource. In the case membersContentAccessRightID is not provided the access right defined for the group resource shall be used. Upon successful validation, the group hosting SCL shall obtain the URIs of all member resources from the attribute *members* of the addressed <*group*> resource and fan out requests addressing the obtained URIs (appended with the relative URI if any) to the member hosting SCLs as indicated in Figure 9.76. If the group hosting SCL determines that multiple member resources belong to one SCL according to the URIs of the member resources, it may converged the requests accordingly before sending out. This may be accomplished by the group hosting SCL creating a <*group*> resource on the member hosting SCL to collect all the members on that member hosting SCL. After receiving the responses from the member hosting SCLs, the group hosting SCL shall then respond to the issuer with the aggregated results and the associated <*memberId*>s.

Member Hosting SCLs: shall treat the request received from the group hosting SCL as a normal CREATE request on the addressed resource as if it comes from the original issuer. Therefore the member hosting SCL shall check if the original issuer has the CREATE permission on the addressed resource. Upon successful validation, the member hosting SCL shall perform the create procedures for the corresponding type of addressed resource as described in other subclauses of clause 9.3.2 and shall send the corresponding response to the group hosting SCL.

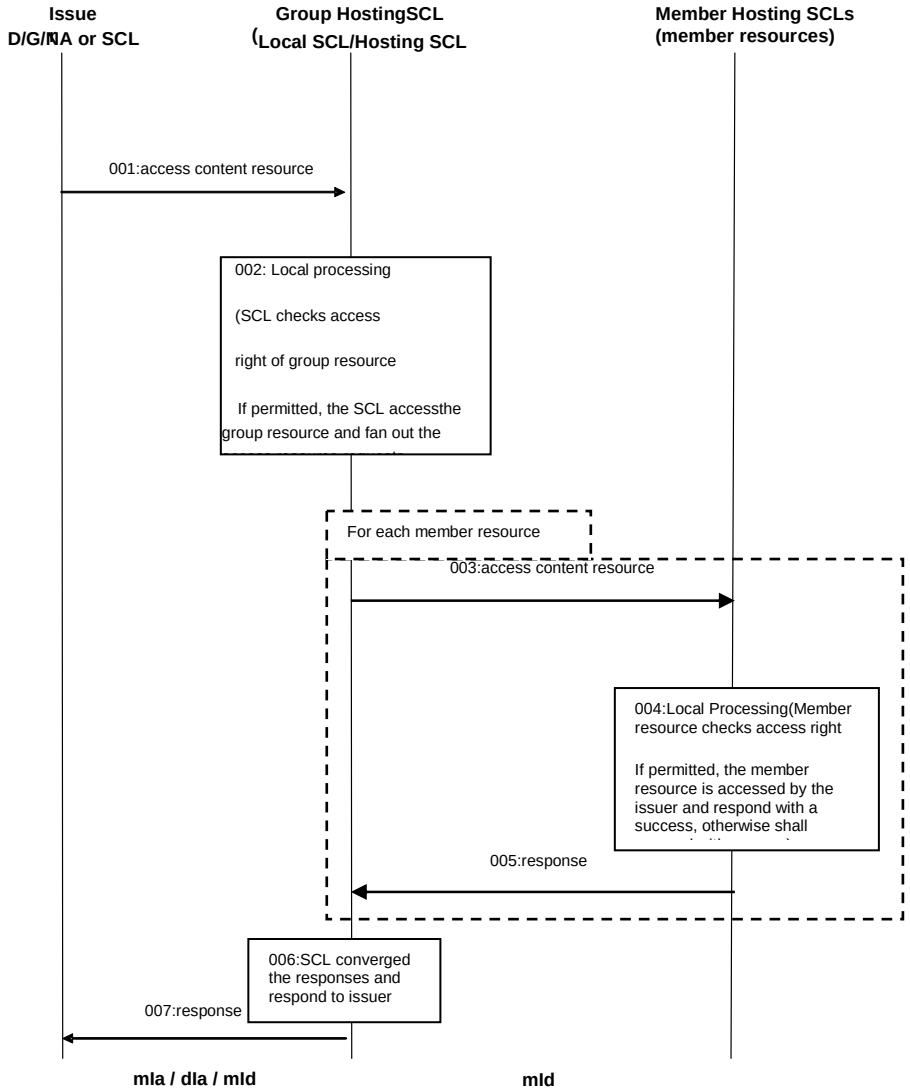


Figure 9.76: Group content management procedures

The procedures illustrated in Figure 9.76 apply to clauses 9.3.2.17.12 to 9.3.2.17.16:

Step 001: To access the content of member resources belonging to an existing <*group*> resource, the issuer shall send a request to an SCL.

Step 002: If the operation is allowed by group hosting SCL, the requests shall be fanned out to the member hosting SCLs for each member resource. If member resources belonging to multiple D' devices locates in the same gateway, the requests may be converged accordingly before sending out to the member hosting SCLs.

Step 003: The requests are fanned out to the member hosting SCLs.

Step 004: If the operation is allowed by the member hosting SCLs, the member resources shall be accessed according to the requested operation (C/R/U/D).

Step 005: The member hosting SCLs shall respond to the group hosting SCL with the appropriate generic responses including any information to be returned according to the requested operation (C/R/U/D).

Step 006: The group hosting SCL shall converge the responses from all the member hosting SCLs.

Step 007: The group hosting SCL shall respond to the issuer with the appropriate responses.

List of main procedure specific exceptions:

Step 002: Issuer is not registered.

Step 002: Issuer does not have the authorization to access the resource.

Step 002: Group resource does not exist.

Step 004: Issuer does not have the authorization to access the member resources.

Step 004: Member resource does not exist.

9.3.2.17.13 Retrieve membersContent

This procedure shall be used for retrieving the content of all member resources belonging to an existing <*group*> resource.

Issuer: shall request to obtaining the content or specific information (e.g. attributes) of all member resources belonging an existing <*group*> resource by using a RETRIEVE verb. The request may address the virtual sub-resource *membersContent* of the specific <*group*> resource of a group hosting SCL for retrieving the content of all member resources. The request may also address the URI that results from appending a relative URI to the *membersContent* URI in order to retrieve the corresponding attributes or sub-resources represented by the relative URI with respect to all member resources. The issuer may be an application or SCL.

Group Hosting SCL: shall check if the issuer has READ permission in the accessRight resource referenced by the membersContentAccessRightID in the addressed <*group*> resource. In the case membersContentAccessRightID is not provided the access right defined for the group resource shall be used. Upon successful validation, the group hosting SCL shall obtain the URIs of all member resources from the *members* attribute of the addressed <*group*> resource and fan out requests addressing the obtained URIs (appended with the relative URI if any) to the member hosting SCLs as indicated in Figure 9.76. If the group hosting SCL determines that multiple member resources belong to one SCL according to the URIs of the member resources, it may converged the requests accordingly before sending out. This may be accomplished by the group hosting SCL creating a <*group*> resource on the member hosting SCL to collect all the members on that member hosting SCL. After receiving the responses from the member hosting SCLs, the group hosting SCL shall then respond to the issuer with the aggregated results and the associated <*memberId*>s.

Member Hosting SCLs: shall treat the request received from the group hosting SCL as a normal RETRIEVE request on the addressed resource as if it comes from the original issuer. Therefore the member hosting SCL shall check if the original issuer has the READ permission on the addressed resource. Upon successful validation, the member hosting SCL shall perform the retrieve procedures for the corresponding type of addressed resource as described in other subclauses of clause 9.3.2 and shall send the corresponding response to the group hosting SCL.

9.3.2.17.14 Update membersContent

This procedure shall be used for updating the content of all member resources belonging to an existing <*group*> resource.

Issuer: shall request to update the content of all member resources belonging to an existing <*group*> resource with the same new data by using a UPDATE verb. The request may address the virtual sub-resource *membersContent* of the specific <*group*> resource of a group hosting SCL to update all member resources. The request may also address the URI that results from appending a relative URI to the "*membersContent*" in order to update only the corresponding attributes or sub-resources represented by the relative URI with respect to all member resources. The issuer may be an application or SCL.

Group Hosting SCL: shall check if the issuer has WRITE permission in the accessRight resource referenced by the membersContentAccessRightID in the group resource. In the case membersContentAccessRightID is not provided the access right defined for the group resource shall be used. Upon successful validation, the group hosting SCL shall obtain the URIs of all member resources from the attribute *members* of the addressed <*group*> resource and fan out requests addressing the obtained URIs (appended with the relative URI if any) to the member hosting SCLs as indicated in Figure 9.76. If the group hosting SCL determines that multiple member resources belong to one SCL according to the URIs of the member resources, it may converged the requests accordingly before sending out. This may be accomplished by the group hosting SCL creating a <*group*> resource on the member hosting SCL to collect all the members on that member hosting SCL. After receiving the responses from the member hosting SCLs, the group hosting SCL shall then respond to the issuer with the aggregated results and the associated <*memberId*>s.

Member Hosting SCLs: shall treat the request received from the group hosting SCL as a normal UPDATE request on the addressed resource as if it comes from the original issuer. Therefore the member hosting SCL shall check if the original issuer has the UPDATE permission on the addressed resource. Upon successful validation, the member hosting SCL shall perform the update procedures for the corresponding type of addressed resource as described in other subclauses of clause 9.3.2 and shall send the corresponding response to the group hosting SCL.

9.3.2.17.15 Delete membersContent

This procedure shall be used for deleting the content of all member resources belonging to an existing <*group*> resource.

Issuer: shall request to delete the content of all member resources belonging to an existing <*group*> resource by using a DELETE verb. The request may address the virtual sub-resource *membersContent* of the specific <*group*> resource of a group hosting SCL to delete all member resources. The request may also address the URI that results from appending a relative URI to the "*membersContent*" in order to delete only the corresponding attributes or sub-resources represented by the relative URI with respect to all member resources. The issuer may be an application or an SCL.

Group Hosting SCL: shall check if the issuer has WRITE permission in the accessRight resource referenced by the membersContentAccessRightID in the <*group*> resource. In the case membersContentAccessRightID is not provided the access right defined for the group resource shall be used. Upon successful validation, the group hosting SCL shall obtain the URIs of all member resources from the attribute *members* of the addressed <*group*> resource and fan out requests addressing the obtained URIs (appended with the relative URI if any) to the member hosting SCLs as indicated in Figure 9.76. If the group hosting SCL determines that multiple member resources belong to one SCL according to the URIs of the member resources, it may converged the requests accordingly before sending out. This may be accomplished by the group hosting SCL creating a <*group*> resource on the member hosting SCL to collect all the members on that member hosting SCL. After receiving the responses from the member hosting SCLs, the group hosting SCL shall then respond to the issuer with the aggregated results and the associated <*memberId*>s.

Member Hosting SCLs: shall treat the request received from the group hosting SCL as a normal DELETE request on the addressed resource as if it comes from the original issuer. Therefore the member hosting SCL shall check if the original issuer has the DELETE permission on the addressed resource. Upon successful validation, the member hosting SCL shall perform the delete procedures for the corresponding type of addressed resource as described in other subclauses of clause 9.3.2 and shall send the corresponding response to the group hosting SCL.

9.3.2.17.16 Subscribe/Un-Subscribe *membersContent*

This procedure shall be used for receiving information about modifications of all member resources belonging to an existing <*group*> resource.

Issuer: shall request to create a subscription resource under all member resources belonging to an existing <*group*> resource by using a CREATE verb. The request shall address the *subscriptions* sub-resource of the virtual sub-resource *membersContent* of the specific <*group*> resource of a group hosting SCL to subscribe to the modifications of all member resources. The request shall include the required information and may include the optional information as described in subscription management clause 9.3.2.19. The issuer may be an application or an SCL.

Group Hosting SCL: shall check if the issuer has WRITE permission in the accessRight resource referenced by the membersContentAccessRightID in the group resource. In the case membersContentAccessRightID is not provided the access right defined for the group resource shall be used. Upon successful validation, the group hosting SCL shall obtain the URIs of all member resources from the attribute *members* of the addressed <*group*> resource and fan out requests addressing the obtained URIs appended with "/subscriptions" to the member hosting SCLs as indicated in Figure 9.76. If the group hosting SCL determines that multiple member resources belong to one SCL according to the URIs of the member resources, it may converged the requests accordingly before sending out. This may be accomplished by the group hosting SCL creating a <*group*> resource on the member hosting SCL to collect all the members on that member hosting SCL. After receiving the responses from the member hosting SCLs, the group hosting SCL shall then respond to the issuer with the aggregated results and the associated <*memberId*>s.

Member Hosting SCLs: shall treat the request received from the group hosting SCL as a normal SUBSCRIBE request on the addressed member resource as if it comes from the original issuer. Therefore the member hosting SCL shall check if the original issuer has the READ permission on the member resource and the CREATE permission on the *subscriptions* sub-resource of the member resource. Upon successful validation, the member hosting SCL shall perform the subscribe procedures for the corresponding type of member resource as described in other subclauses of clause 9.3.2 and shall send the corresponding response to the group hosting SCL.

9.3.2.18 Subscriptions collection management

9.3.2.18.1 Introduction

This clause describes different procedures for managing the collection resource *subscriptions* as defined in clause 9.2.3.22. Such type of resource (like any other collection) cannot be created or deleted by means of a request over the reference points: mIa, mId and dIa.

9.3.2.18.2 Retrieve *subscriptions*

This procedure shall be used for getting all attributes of the *subscriptions* collection resource and the list of reference to all <*subscription*> children resources, for which the Issuer is authorized to discover. This procedure is identical to the one specified in clause 9.3.2.16.2 with this exception: the *subscriptions* collection does not have an accessRight reference. Therefore, the hosting SCL shall only allow retrieval if the issuer has delete permission on the parent resource.

9.3.2.18.3 Update *subscriptions*

This procedure shall be used for modifying all or some of the attributes defined in the *subscriptions* collection resource.

The procedure is described in details in clause 9.3.2.30.3.

9.3.2.18.4 Subscribe/Un-Subscribe to *subscriptions*

This operation is not applicable.

9.3.2.19 Subscription management

9.3.2.19.1 Introduction

A subscriber (see note) (an Application or an SCL) may ask to be notified when resources are modified. A subscription is the relation between the subscriber and the Hosting-SCL of the Subscribed-to Resource. The Hosting SCL shall notify the subscriber of any changes in the Subscribed-to Resource under the conditions provided when the subscription was created or modified.

NOTE: In this clause the term subscriber and Issuer are used interchangeably.

A subscription shall be represented by a resource itself. This allows manipulation of the subscription in a resource oriented manner, e.g. the conditions of a subscription may be modified by modifying the <*subscription*> resource or parts thereof, or a subscriber may unsubscribe by deleting the subscription resource.

The procedures below depict the creation, modification and deletion of the subscription resource as well as the procedures followed by the hosting SCL on modifications of the subscribed-to resource.

9.3.2.19.2 Create <*subscription*> (Subscribe for modifications to a resource)

This procedure shall be used to subscribe for modifications to a resource (the Subscribed-to Resource). After a modification of the Subscribed-to Resource, its new resource representation is sent to the subscriber in a notify message.

Issuer: requests to create a subscription resource by using CREATE. The issuer shall be the subscriber. The request shall address a *subscriptions* collection resource child of the Subscribed-to Resource: <*resourceURI*>*/subscriptions*. The resource shall be addressed as defined in clause 9.2.3.23.

Resources that cannot be subscribed to, shall not have such a *subscriptions* collection child. Therefore, a subscription to these resources shall fail with a "not found" error response. See clause 9.2.3.23 for details on the resource structure.

The subscriber shall request to receive notifications of modifications of the Subscribed-to Resource. The request may include a *contactURI*, a filter criteria and an *expirationTime*. It may also include the *minimalTimeBetweenNotifications* or a *delayTolerance* (by default neither of these). The request may also indicate if the subscriber is interested changes in the complete resource or only in a certain attribute of the resource.

After the successful creation of a <*subscription*> resource, the subscriber shall receive notifications of modifications in the Subscribed-to Resource.

Hosting SCL: shall check if the request is valid and shall check if the issuer (the subscriber) has the proper accessRights for creating a subscription. If the issuer is allowed to read the Subscribed-to Resource, it shall also have the permission to create a subscription resource in its child *subscriptions* collection resource.

The Hosting SCL may modify some of the attributes provided in the request. Also the hosting SCL shall provide default values for mandatory not provided attributes and shall provide values for the read-only attributes of the <*subscription*> resource. The Hosting SCL shall then create a <*subscription*> resource using the provided (but optionally changed) attribute values. In case of success, the hosting SCL shall include the URI of the created <*subscription*> resource in the response message (to be used by the issuer to remove or modify the subscription).

After the creation of the <*subscription*> resource, the Hosting SCL shall return a generic success response.

In addition to generating a success response, the Hosting SCL shall also send a notification with the current state of the resource (see clause 9.3.2.19.6 on notification). The notification shall only be sent if the filterCriteria match. Otherwise, the first notification shall be sent when the Subscribed-to Resource is modified in such a way that the filterCriteria match.

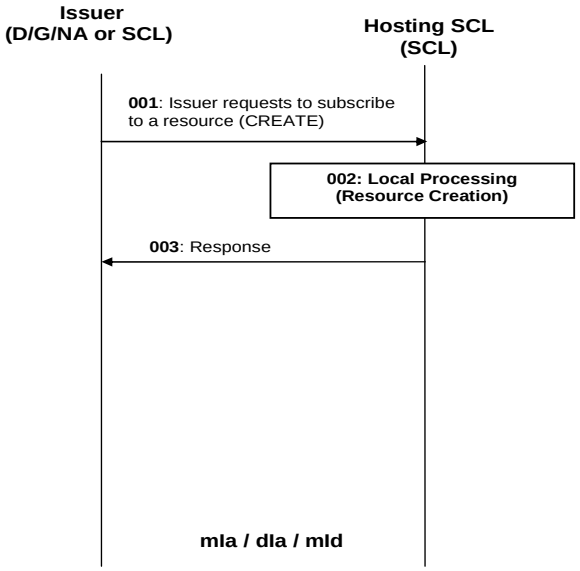


Figure 9.77: Procedures for resource subscription

Step 001: The Issuer (Subscriber) shall send a CREATE request to the Hosting SCL (Subscribe-To SCL). For more details on the request see text above.

Step 002: The Hosting SCL shall check if the Issuer is authorized to create the resource for the subscription. The SCL includes the characteristics of the resource to be created. Characteristics can contain information like the expiration time. More details about the behaviour of the Hosting SCL are provided above.

Step 003: The Hosting SCL responds positively to the request.

List of main procedure specific exceptions:

Step 002: The provided characteristics are not acceptable to Hosting SCL. The Hosting SCL responds with an error.

9.3.2.19.3 Update <*subscription*>

This procedure shall be used to update a <subscription> resource. Alternatively, the issuer can request to update only a specific attribute or part of an attribute, as described in partial addressing, see clause 9.3.2.29.

Reasons to change a subscription may include the extension of its lifetime (if the subscription resource expires the subscription will end) or the modification of the *contactURI*, *minimalTimeBetweenNotifications* or *delayTolerance*. It shall not be allowed to modify some aspects of the subscription, such as filterCriteria.

Issuer: requests to update a subscription resource. The request shall address a <*subscription*> resource as defined in clause 9.2.3.23, the issuer may send new (proposed) values for all mandatory read-write attributes and maysend values for the optional read-write attributes.

Hosting SCL: shall check if the request is valid and shall check if the issuer is the original subscriber (the only entity that shall have access right for the subscription). The Hosting SCL may ignore some provided attributes values in the request. Also the Hosting SCL may modify some of the attributes provided in the request before modifying the <*subscription*> resource, e.g. it may shorten a suggested expiration time. If the complete resource is modified, the new values shall be regarded as a complete replacement of all the attributes, however the Hosting SCL shall not accept value changes to attributes that may not be modified for a subscription (such as the filterCriteria). The hosting SCL shall then return a generic success response.

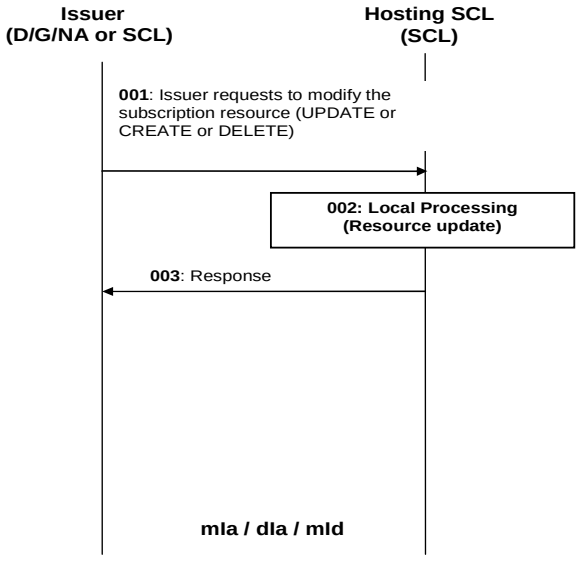


Figure 9.78: Procedures for subscription update

Step 001: The Issuer requests the Hosting SCL to modify a subscription resource. The Issuer has three options for updating the resource as indicated in the text above. The Issuer can therefore use an UPDATE, CREATE or a DELETE.

Step 002: The Hosting SCL shall check if the Issuer is authorized to perform the modification to the subscription resource. The SCL shall perform the changes to the resource. More details about the behaviour of the Hosting SCL are provided above.

Step 003: The Hosting SCL responds positively to the request.

List of main procedure specific exceptions:

Step 002: The Issuer is not authorized to modify (update, create or delete) the resource. The Hosting SCL responds with an error.

9.3.2.19.4 Retrieve <subscription>

This procedure shall be used to retrieve a subscription. This shall return information about the subscription such as the *expirationTime*, *filterCriteria*, *contactURI,* etc. Alternatively, the issuer can request to retrieve only a specific attribute or part of an attribute, as described in partial addressing, see clause 9.3.2.29.

Issuer: requests to retrieve a <*subscription*> resource (using the Retrieve method). The request shall address a subscription resource. The resource is addressed as defined in clause 9.2.3.23.

Hosting SCL: Shall check if the request is valid and shall authorize the request. If the issuer is the subscriber (i.e., the creator of the <*subscription*> resource) or if the issuer is allowed to delete the Subscribed-to Resource, it shall also have the permission to retrieve a <*subscription*> resource.. The Hosting SCL shall return the complete representation of the subscription resource or the value of a specific attribute of the subscription resource depending on how the resource was addressed.

It shall return a generic success response.

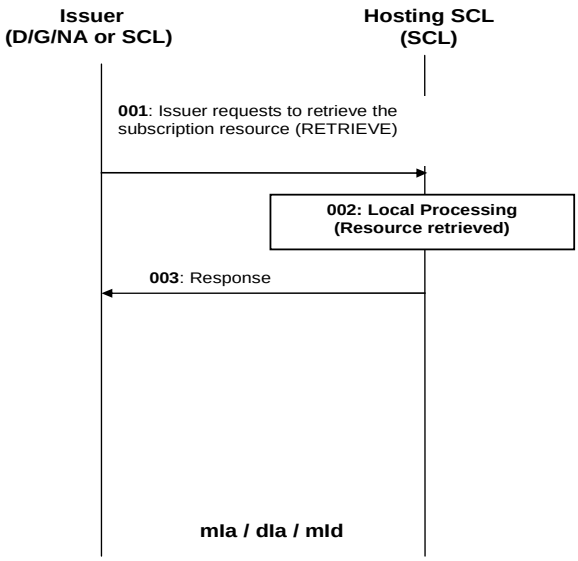


Figure 9.79: Procedures for subscription retrieve

Step 001: The Issuer requests the Hosting SCL to retrieve a subscription resource, as indicated in the text above.

Step 002: The Hosting SCL shall check if the Issuer is authorized to retrieve the subscription resource. More details about the behaviour of the Hosting SCL are provided above.

Step 003: The Hosting SCL responds positively to the request.

List of main procedure specific exceptions:

Step 002: The Issuer is not authorized to retrieve the resource. The Hosting SCL responds with an error.

9.3.2.19.5 Delete <*subscription*> (Unsubscribe)

This procedure may be used to delete an active subscription, which means that the subscriber unsubscribes from notifications. This is just one of the ways in which subscriptions can end. Other ways include returning error responses on notifications or letting the subscription expire without refreshing it.

Issuer: requests to delete a subscription resource, using the DELETE verb. The request shall address a <*subscription*> resource. The resource shall be addressed as defined in clause 9.2.3.23.

Hosting SCL: shall check if the request is valid and shall check if the issuer has the proper accessRights for creating a subscription. Only the original subscriber shall have the right to delete the subscription resource. The deletion shall not lead to a notification of the subscriber.

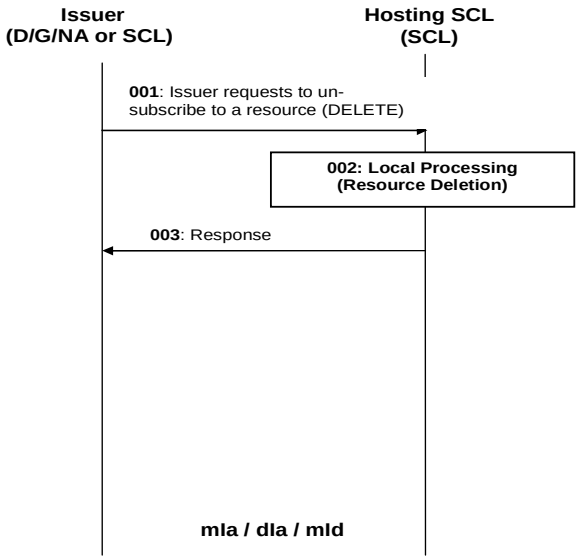


Figure 9.80: Procedures for un-subscribe

Step 001: The Issuer sends a DELETE request to the Hosting SCL, as described above.

Step 002: The Hosting SCL shall check if the Issuer is authorized to delete the subscription resource. The SCL shall remove the resource. More details about the behaviour of the Hosting SCL are provided above.

Step 003: The Hosting SCL responds positively to the request.

List of main procedure specific exceptions:

Step 003: The Issuer is not authorized to delete the resource. The Hosting SCL responds with an error.

9.3.2.19.6 Notification

This procedure shall be used to notify the subscriber of a modification of a resource for which it has an active subscription.

Hosting SCL: The procedure starts when the hosting SCL detects a modification of the subscribed-to Resource that matches the specified filterCriteria as specified in the <*subscription*> resource.

The Hosting SCL shall check the *delayTolerance* of the subscription and the *minimalTimeBetweenNotifications* attribute. Only one or none of these two attributes shall be present in the subscription.

The *minimalTimeBetweenNotifications* determines when the next notification can be sent and together with the last time a notification was sent for this subscription determines the *earliest* time a new notification may be sent (say t1).

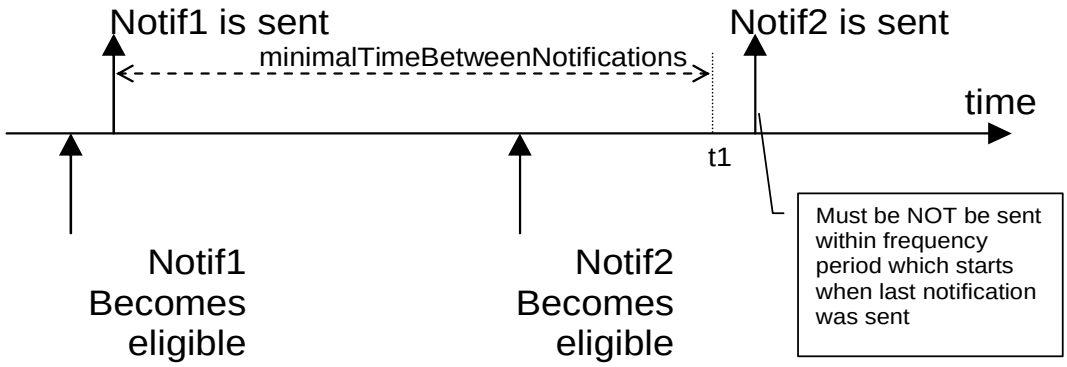


Figure 9.81: Notification mechanism when minimalTimeBetweenNotifications is used

If a new notification becomes eligible before the previous notification was delivered, the previous notification shall be ignored. There shall be only one notification per subscription pending or active at any one time. This means that the subscriber might receive only one notification after there have been multiple modifications to the Subscribed-to Resource.

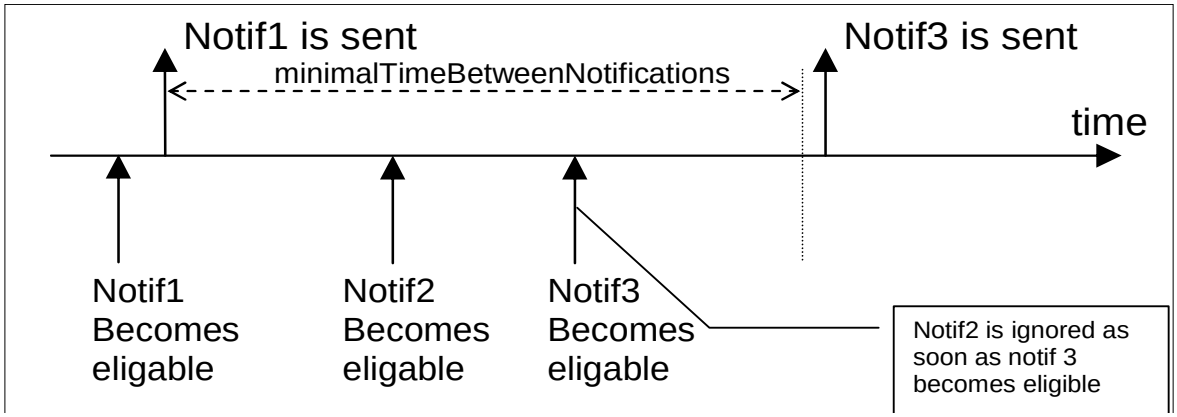


Figure 9.82: Notification mechanism when minimalTimeBetweenNotifications is used, the case when more than one notification become eligible

The *delayTolerance* determines how urgent the notification is, i.e. the *latest* time a notification shall be sent (say t2). The *delayTolerance* of a certain notification is determined by the *delayTolerance* attribute of the subscription and the *delayTolerance* attribute of the resource instance creation that generated the notification. If both are applicable, the shorter of the two intervals shall determine the *delayTolerance* of the notification.

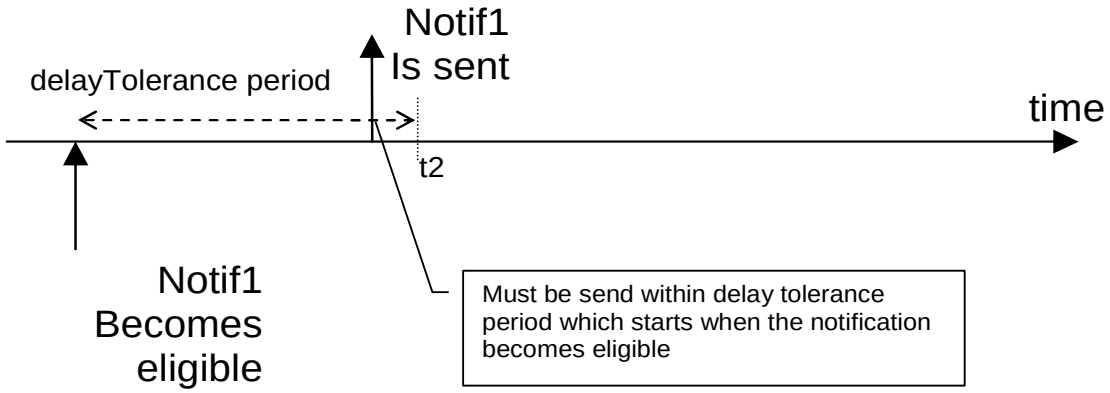


Figure 9.83: Notification mechanism when delayTolerance is used

Conflict resolution

This clause describes when the Hosting SCL shall send the notification, especially since there can be conflicts between the various facets.

Multiple *delayTolerance*

In case of multiple (overlapping) *delayTolerance* periods, the original delayTolerance remains in effect. This means that *delayTolerance* period of the replaced notification is still in effect.

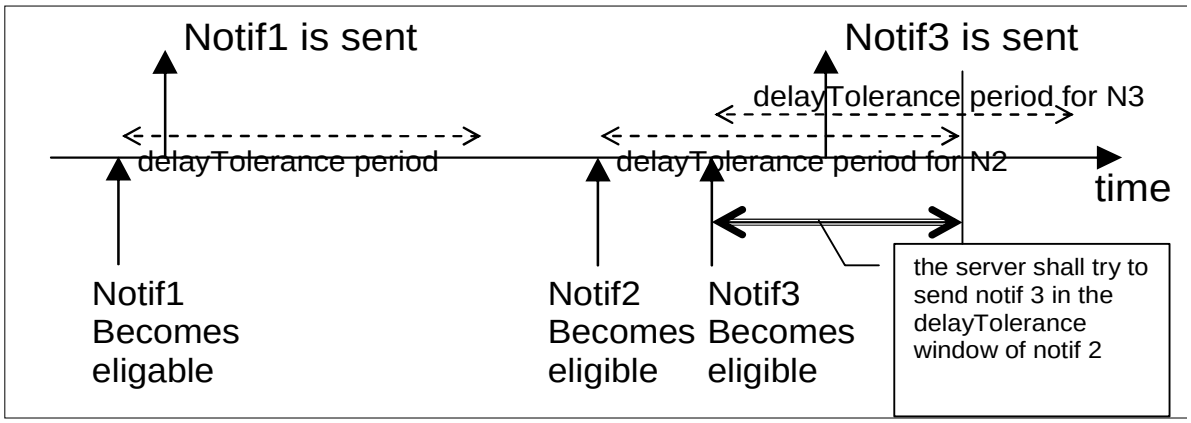


Figure 9.84: Notification mechanism, conflict resolution when multiple delayTolerance are used

Only if the last notification cannot be delivered within the original *delayTolerance* period, it shall be delivered in the *delayTolerance* period associated with the last notification itself. If the notification cannot be delivered in the *delayTolerance* period of the associated event, then the Hosting SCL may keep retrying to send the notification according to SCL policies until either a new event becomes eligible or until the Hosting SCL decides to remove the subscription.

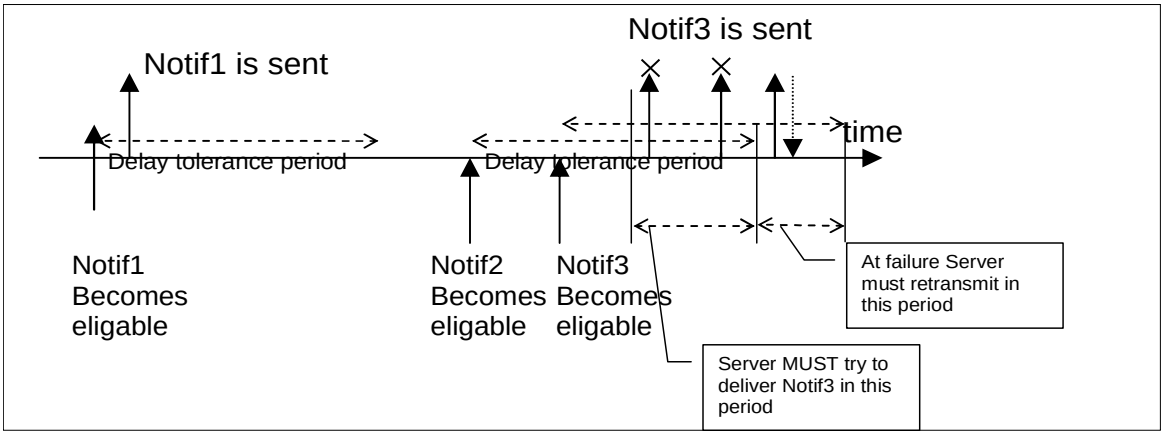


Figure 9.85: Notification mechanism, the case of failure to transmit notifications on time

No conflict

There is no conflict if t1<t2. In such a case the SCL shall send the notification after the current *minimumTimeBetweenNotifications* period expires and before the *delayTolerance* deadline.

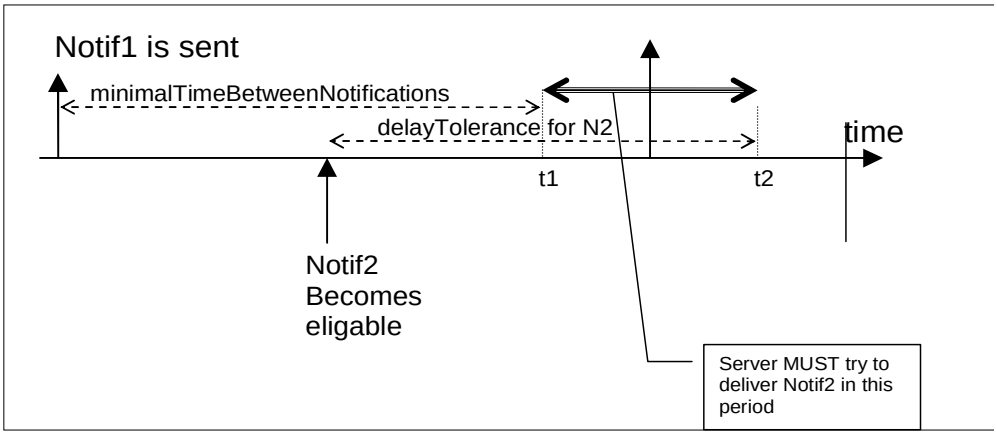


Figure 9.86: Notification mechanism, no conflict between the subscription minimalTimeBetweenNotification and the resource instance delayTolerance

Conflict between *delayTolerance* and *minimalTimeBetweenNotifications*

There is a conflict if t2<t1. In such a case the SCL shall ignore the *minimalTimeBetweenNotifications* and shall send the notification within the *delayTolerance* period.

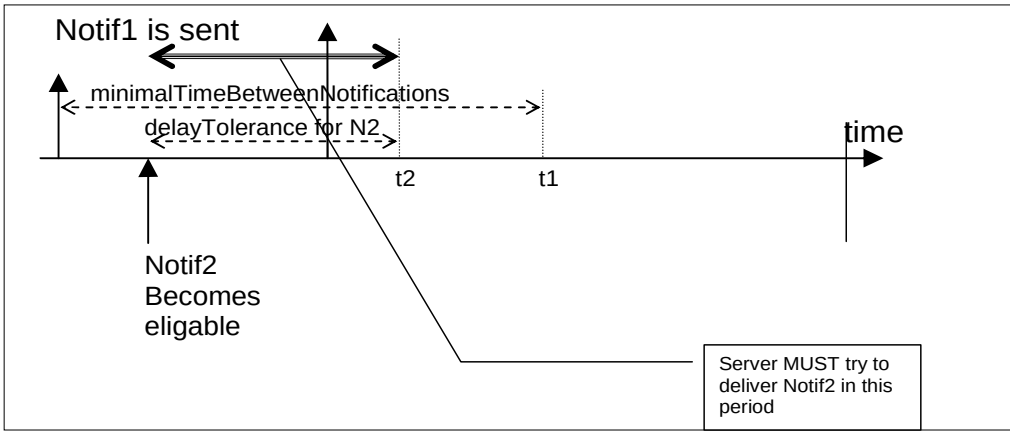


Figure 9.87: Notification mechanism, conflict between the subscription minimalTimeBetweenNotification and the resource instance delayTolerance

The hosting SCL shall include a notification structure as shown below and shall send this to the contactURI specified in the subscription.

Table 9.62

Attribute	Mandatory/Optional	Description
statusCode	M	The status code, that would correspond to a retrieve request on the Subscribed-to Resource. In case of partial access (see clause 9.3.2.29) this corresponds to the retrieve using the attribute accessor from the filter-criteria. If the status code indicates an error, then this also indicates that the subscription resource has been deleted.
representation	O	The representation of the Subscribed-to Resource in case the status code indicated "success". For status codes indicating errors, the representation may contain additional error information that would have been present in the corresponding retrieve response to the Subscribed-to Resource.
subscription	M	Reference to the <*subscription*> resource that generated this notification.

Note that the notification does not represent a REST resource, but more like an asynchronous response to a RETRIEVE packaged in a NOTIFY request.

In case of asynchronous subscriptions, the *contactURI* is used to directly send the information to the subscriber. In case of long-polling, see clause 9.3.2.26.6.

For notifications that are sent to a *contactURI*, there will be a response. A response to the notification indicating an error actively issued by the subscriber shall result in the termination of the subscription and removal of the <*subscription*> resource. Responses that indicate indirect failures, may result in retries of the notification. The number of retries, when the retries are performed etc, is a policy defined in the Hosting SCL.

If the Hosting SCL can determine that the subscriber is offline, the notification shall be delayed until the subscriber becomes available again, if this falls within its delay-tolerance period.

If the hosting SCL can determine that the subscriber will be unavailable for a longer time than indicated by the *delayTolerance* (e.g. by checking the schedule parameter on the registration or based on heuristic analysis of the past notifications), it may attempt to force the subscriber to become available ahead of schedule. The mechanism to request the device to become available is access network dependent (e.g. it may be based on an SMS) and is outside the scope of the present document.

If the hosting SCL fails to send the notification, then according to server policy (e.g. based on the number of retries or the period in which notifications fail) the hosting SCL may change the status of the <*scl*> registration resource to NOT_REACHABLE.

If the *contactURI* represents a <*container*> resource located in the hosting SCL, then the hosting SCL shall add all notifications as content instances to the container, even if multiple notifications become eligible within the *delayTolerance* period of the first notification. This ensures, that such a setup can be used to "buffer" all notifications, and can be used if, for example, the subscriber expects connection problem or knows that it goes off-line and still wants to receive every notification that happened during that period (within the parameters of the container).

Subscriber: The subscriber receives the notification and sends a successful response back. In case it wants to terminate the subscription it may send an error response on the notification request as indicated above.
The sending of responses to a notification only applies to the asynchronous subscriptions.

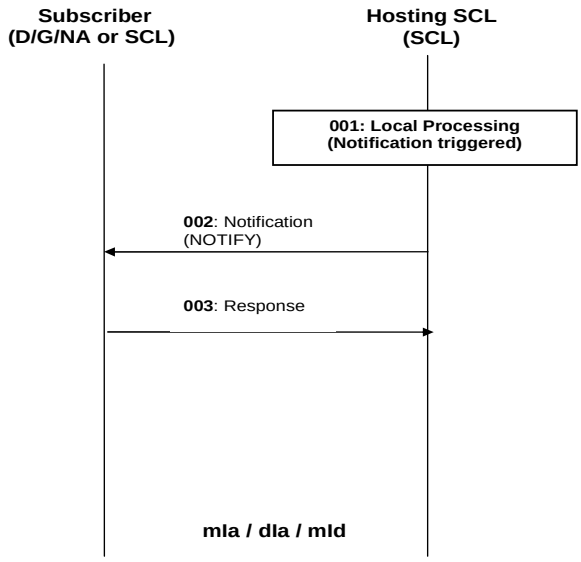


Figure 9.88: Procedures for asynchronous notification

Step 001: The Hosting SCL identifies an event that needs to be reported to the subscriber.

Step 002: The Hosting SCL shall provide the notification to the subscriber. More details about the behaviour of the Hosting SCL are provided above.

Step 003: The Issuer responds positively to the request.

9.3.2.19.7 Deletion of the Subscribed-to Resource

This procedure is used to notify the subscriber of a deletion of the subscribed resource.

Hosting SCL: if the resource that is being subscribed to is deleted all the subscribers in the child *subscriptions* collection resource shall be notified.

Subscriber: The subscriber receives the notification and sends a response back. Any further attempt to access the subscription resource will fail.

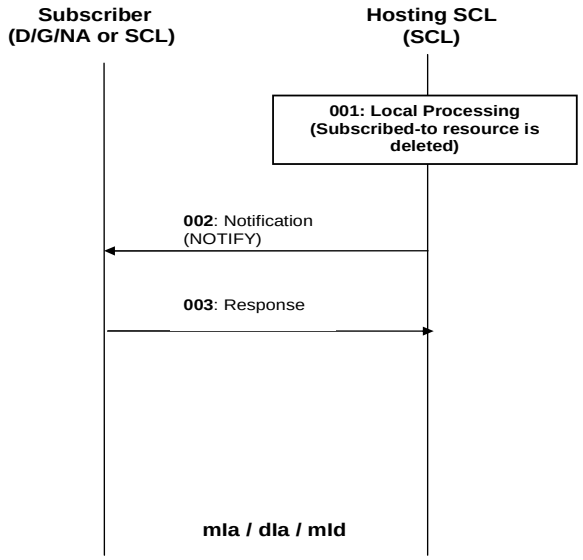


Figure 9.89: Procedures for asynchronous notification in case of deletion of subscribed resource

Step 001: The Hosting SCL identifies that the subscribed-to resource has been deleted and an event that needs to be reported to the subscriber.

Step 002: The Hosting SCL shall provide the notification to the subscriber. More details about the behaviour of the Hosting SCL are provided above.

Step 003: The Issuer responds positively to the request.

9.3.2.20 M2M Pocs Collection management

9.3.2.20.1 Introduction

This clause describes different procedures for managing the collection resource *m2mPocs* as defined in clause 9.2.3.24. Such type of resource (like any other collection) cannot be created or deleted by means of a request over the reference points: mIa, mId and dIa.

9.3.2.20.2 Retrieve *m2mPocs*

This procedure is used for getting a list of references to all *m2mPocs* in the addressed collection resource.

The procedure is described in details in clause 9.3.2.30.2.

9.3.2.20.3 Update *m2mPocs*

This procedure is used for updating a list of references to all *m2mPocs* in the addressed collection resource.

The procedure is described in details in clause 9.3.2.30.3.

9.3.2.20.4 Subscribe/Un-Subscribe to *m2mPocs*

This operation is not applicable.

9.3.2.21 M2M PoC management

9.3.2.21.1 Introduction

This clause describes different procedures for managing the resource <*m2mPoc*> as defined in clause 9.2.3.25.

9.3.2.21.2 Create <*m2mPoC*>

The procedure is used for adding an <*m2mPoc*> to the *m2mPocs* collection resource in an SCL, i.e. this occurs when the device attaches to a new access network.

Issuer: shall request to create a new <*m2mPoc*> resource using the CREATE verb. The request shall address a URI of *m2mPocs* collection resource. The request may provide the identity of the <*m2mPoc*> resource. The request shall also provide the corresponding contactInfo information and may include an expiration time.

Hosting SCL: shall check the permissions for creating the resource, validity of provided attributes, as well as the existence of the addressed collection resource. Then it shall then create a new <*m2mPoc*> resource.

The creation shall only be allowed if the requestor of the creation is the creator of the parent <*scl*> resource, i.e. only the registered SCL can create its own <*m2mPoc*> resource.

If a resource with the provided name already exists, or if no identity is provided, the hosting SCL shall provide a new and unique identity. Then the hosting SCL shall send a response to the issuer. The hosting SCL shall also provide in the success response the URI of the m2mPoc resource with identity <*m2mPoc*>.

If an <*m2mPoc*> is added with its *onlineStatus* set to ONLINE and the current online status of the SCL registration is OFFLINE or NOT_REACHABLE, the hosting SCL shall change the *onlineStatus* of the issuer SCL registration resource to ONLINE.

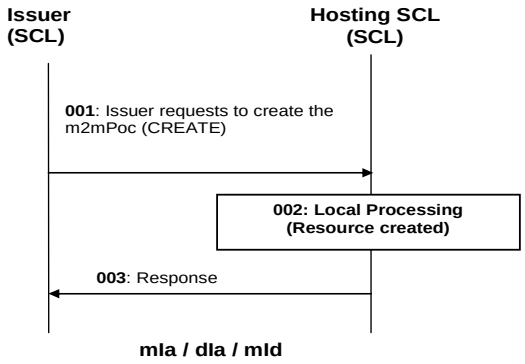


Figure 9.90: Procedures for <*m2mPoc*> create

Step 001: The Issuer sends the request to the Hosting SCL to create a <*m2mPoc*> resource.

Step 002: The Hosting SCL shall check if the Issuer is authorized to create the <*m2mPoc*> resource.

Step 003: The Hosting SCL responds positively to the request with the reference to the URI of the <*m2mPoc*> resource.

List of main procedure specific exceptions:

Step 002: The Issuer is not authorized to create the resource. The Hosting SCL responds with an error.

9.3.2.21.3 Retrieve <*m2mPoC*>

The procedure is used for retrieving any of the existing information stored in any of the attributes that compose an <*m2mPoc*> resource.

Issuer: shall request to retrieve all or any of the attributes of the existing <*m2mPoc*> resource by using RETRIEVE. The request is addressing the specific "<*m2mPoc*>" resource of the Receiver SCL.

The retrieval is only allowed if the requestor of the retrieval is the creator of the parent <*scl*> resource or the hosting SCL, i.e. only the registered SCL and the hosting/registered-to SCL can retrieve an <*m2mPoc*> resource.

Hosting SCL: after checking the existence of the addressed <*m2mPoc*> resource and the permissions for retrieving the resource, the SCL shall return the requested information.

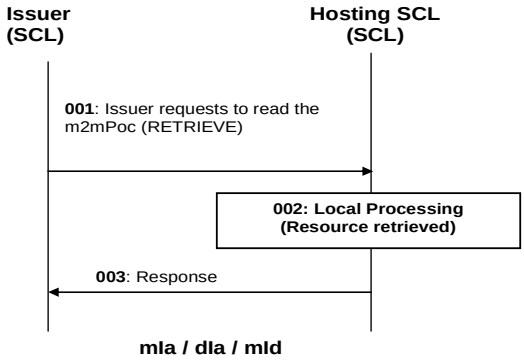


Figure 9.91: Procedures for m2mPoc retrieve

Step 001: The Issuer sends a request to the Hosting SCL to retrieve a m2mPoc resource.

Step 002: The Hosting SCL shall check if the Issuer is authorized to retrieve the m2mPoc resource.

Step 003: The Hosting SCL responds positively to the request with the representation of the m2mPoc resource.

List of main procedure specific exceptions:

Step 002: The Issuer is not authorized to retrieve the resource. The Hosting SCL responds with an error.

9.3.2.21.4 Update <*m2mPoC*>

The procedure is used for updating the <*m2mPoc*> resource. Update any of the existing attributes of an <*m2mPoc*> resource is defined in clause 9.3.2.29.

Updating is typically used when the point of attachment in the access network changes, to refresh an <*m2mPoc*> resource to keep it from expiring or to set the *onlineStatus* of the m2mPoc to indicate whether the m2mPoc can be used for reaching the SCL.

Issuer: shall issue a request to update the <*m2mPoc*> resource, using a Update verb. The request addresses a <*m2mPoc*> as defined in clause 9.2.3.25. The issuer shall send new proposed values for all mandatory read-write attributes and optionally sends values for the optional read-write attributes. They will be handled similarly as in the create request.

Hosting SCL: shall validate the received request. Update shall only be allowed if the requestor is either the registered SCL or the registered-to/hosting SCL.

The hosting SCL shall set default values for not provided mandatory attributes.
The hosting SCL shall set the onlineStatus of the issuer SCL based on the provided information in the *onlineStatus* of the <*m2mPoc*>.

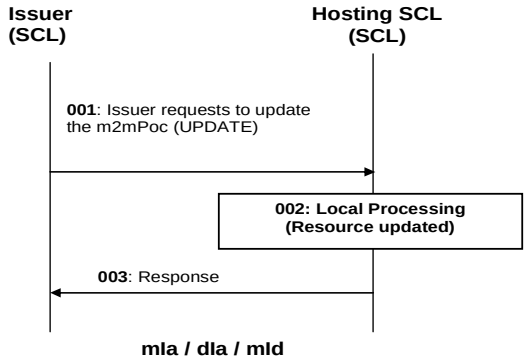


Figure 9.92: Procedures for m2mPoc update

Step 001: The Issuer requests to the Hosting SCL to update an m2mPoc resource.

Step 002: The Hosting SCL shall check if the Issuer is authorized to update the m2mPoc resource.

Step 003: The Hosting SCL responds positively to the request.

List of main procedure specific exceptions:

Step 002: The Issuer is not authorized to update the resource. The Hosting SCL responds with an error.

9.3.2.21.5 Delete <*m2mPoC*>

The procedure is used for deleting an <*m2mPoc*> resource, e.g. this will occur when the device detaches from an access network.

Issuer: shall request to delete an m2mPoc resource using a DELETE verb. The request shall address the specific <*m2mPoc*> resource of the Hosting SCL.

Hosting SCL: shall check the existence of the addressed,<*m2mPoc*>. resource and the permissions for the issuer to delete the resource. The SCL shall remove the m2mPoc resource and return a success response to the issuer.

The deletion is only allowed if the requestor of the deletion is the creator of the parent <*scl*> resource or the hosting SCL, i.e. only the registered SCL and the hosting/registered-to SCL can delete an <*m2mPoc*> resource. The hosting SCL shall set the onlineStatus of the issuer SCL based on the provided information in the *onlineStatus* of the <*m2mPoc*>.

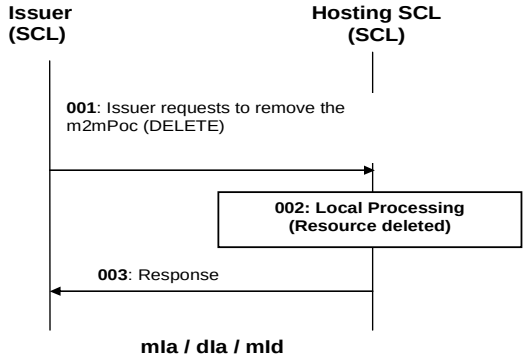


Figure 9.93: Procedures for <*m2mPoc*> delete

Step 001: The Issuer requests to the Hosting SCL to remove an m2mPoc resource.

Step 002: The Hosting SCL shall check if the Issuer is authorized to remove the m2mPoc resource.

Step 003: The Hosting SCL responds positively to the request.

List of main procedure specific exceptions:

Step 002: The Issuer is not authorized to remove the resource. The Hosting SCL responds with an error.

9.3.2.22 Management Objects collection management

9.3.2.22.1 Introduction

This clause describes different procedures for managing the retrieval and updates of information associated with an *mgmtObjs* resource as defined in clause 9.2.3.26. Such type of resource (like any other collection) cannot be created or deleted by means of a request over the reference points: mIa, mId and dIa.

9.3.2.22.2 Retrieve mgmtObjs

This procedure shall be used for getting all attributes of an *mgmtObjs* collection resource and the references to visible child resources.

The procedure is described in details in clause 9.3.2.29.2.

9.3.2.22.3 Update mgmtObjs

This procedure shall be used for modifying the attributes in an *mgmtObjs* collection resource.

The procedure is described in details in clause 9.3.2.23.4.

9.3.2.22.4 Subscribe/Un-Subscribe to *mgmtObjs*

These procedures shall be used for subscribing and un-subscribing for changes in an *mgmtObjs* collection resource.

The procedures are described in details in clause 9.3.2.19.

9.3.2.23 Management Object management

9.3.2.23.1 Introduction

This clause describes the remote entity management procedures over mIa and mId reference points. Different RESTful requests addressing an <*mgmtObj*> resource (or its attributes or sub-resources) shall be translated into existing management commands and procedures performed on the mapped Management Objects (MO) data on the remote entity, based on the converting mechanism described in the following clauses.

9.3.2.23.2 Create <mgmtObj>

The procedure is used to create a specific <*mgmtObj*> resource in the hosting SCL to expose at run-time the corresponding management function of a remote entity (i.e. M2M Device/Gateway) over mIa reference point. Depending on the data model being used, the created <*mgmtObj*> resource may be a partial or complete mapping from the MO on the remote entity. If such MO is missing from the remote entity, it shall be added to the remote entity by this procedure. Further RESTful operations performed on the created <*mgmtObj*> resource shall be converted by the hosting SCL into a corresponding device management action performed on the mapped MO on the remote entity over mId reference point using existing device management protocols (e.g. OMA-DM or BBF TR-069).

Issuer: shall request to create a new <*mgmtObj*> resource to be named as <*mgmtObj*> by using a CREATE verb. The request shall address an *mgmtObjs* collection resource of the hosting SCL as described in clause 9.2.3.26. The request may also provide the attributes as described in clause 9.2.3.27. If the <*mgmtObj*> resource contains one or more <*parameters*> sub-resources which may also contains <*parameters*> sub-resources recursively, the issuer may create each of the <*parameters*> sub-resource gradually as needed by sending the CREATE request addressing the parent resource of <*parameters*> the sub-resource.

The issuer may be:

- The D/GSCLD/GSCL on the remote entity: In this case, the D/GSCLD/GSCL first collects the original MO data (the management tree structure or also the value of the tree nodes if needed) of the local device and transforms the data into the <*mgmtObj*> resource representation, then requests the hosting SCL to create the corresponding <*mgmtObj*> resource.

- An M2M NA: In this case, the NA requests the hosting SCL to add the corresponding MO data to the remote entity by creating an <*mgmtObj*> resource in the hosting SCL.

NOTE 1: The hosting SCL in the network domain may also create the <*mgmtObj*> resource locally by itself. The details are out of scope. In this case, the hosting SCL first collects the original MO data on the remote entity via mId reference point following existing (e.g. OMA-DM or BBF TR-069) device management protocols, then transforms the data into the <*mgmtObj*> resource representation and create the <*mgmtObj*> resource locally in the hosting SCL.

NOTE 2: The <*mgmtObj*> resource may also be created in the hosting SCL by other offline provisioning means which are out of scope.

Hosting SCL: shall check if the issuer has the CREATE permission on the addressed "*mgmtObjs*" resource (or the parent <*mgmtObj*> or <*parameters*> resource in the case of sub-resource creation). The hosting SCL shall also check the validity of provided attributes. Upon successful validation,, a new <*mgmtObj*> resource with name "<*mgmtObj*>" is created in the hosting SCL with the provided attributes. If the issuer is an NA, the hosting SCL shall also send the management request over mId reference point to add the corresponding MO data to the remote entity based on existing management protocol. The hosting SCL shall maintain the mapping relationship between the created <mgmtObj> resource and the original MO data on the remote entity. Then the hosting SCL shall respond to the issuer with the appropriate generic responses as indicated in clause 9.3.1.1. It shall also provide in the response the URL of the created new resource.

The hosting SCL shall be:

- An NSCL.

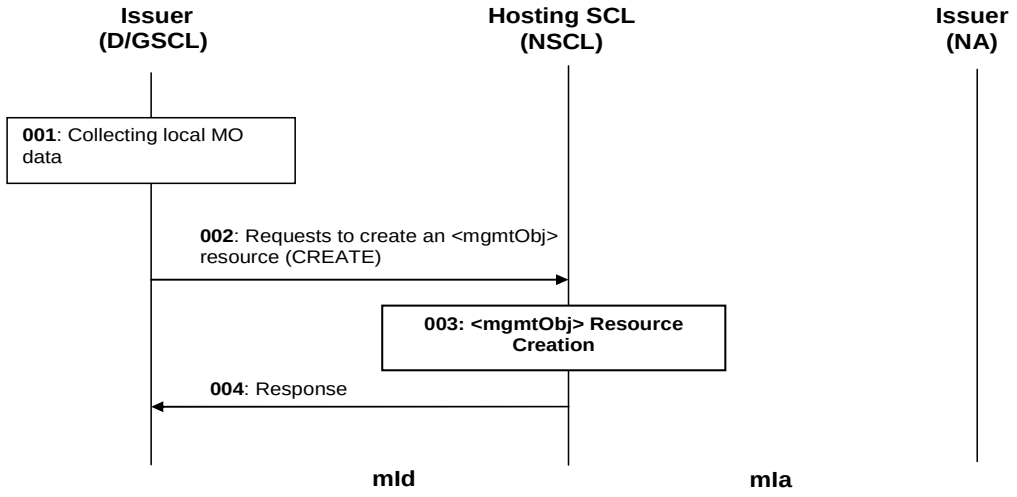


Figure 9.94: Illustration of procedures to create an <*mgmtObj*> resource by D/GSCL

Step 001: D/GSCL shall first collect local MO data to be exposed by the <*mgmtObj*> resource.

Step 002: To create an <*mgmtObj*> resource, D/GSCL shall issue a CREATE request with the collected MO data to the NSCL NSCL.

Step 003: If the creation is allowed, an <*mgmtObj*> resource shall be created in the NSCL with the provided MO data and other required attributes (e.g. "originalMO", "moID").

Step 004: A response shall be returned to D/GSCL.

List of procedure specific exceptions:

Step 003: If the creation is not allowed, the <*mgmtObj*> resource shall not be created, and a proper error code shall be returned to D/GSCL in step 004.

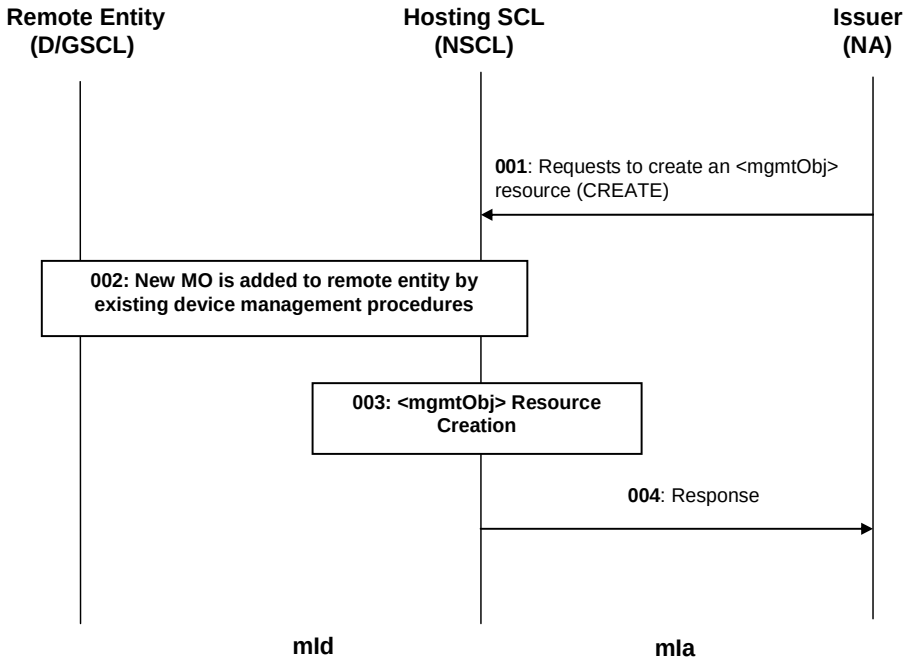


Figure 9.95: Illustration of procedures to create an <*mgmtObj*> resource by NA

Step 001: To add a new MO to a remote entity (D/GSCL), NA shall issue a CREATE request with the MO data to the NSCL to create an <*mgmtObj*> resource. The request shall address the *mgmtObjs* resource under the <*scl*> resource that represents the registered D/GSCL in the NSCL.

Step 002: If the creation is allowed, NSCL shall trigger existing (OMA-DM or BBF TR-069) device management procedures to add a corresponding new MO to the remote entity.

Step 003: An <*mgmtObj*> resource shall be created in the NSCL with the provided MO data and other required attributes (e.g. "originalMO", "moID").

Step 004: A response shall be returned to NA.

List of procedure specific exceptions:

Step 002: If the creation is not allowed, steps 002-003 shall be skipped and a proper error code shall be returned to NA in step 004.

Step 002: If the corresponding MO cannot be added to remote entity due to some reason (e.g. not reachable, memory shortage), step 003 shall be skipped and a proper error code shall be returned to NA in step 004.

Step 002: If the corresponding MO exists already on the remote entity and it is not a multi-instance MO, then no new MO is added to the remote entity. Step 003 shall still be performed and a successful response shall be returned to NA in step 004.

NOTE 3: In the case that asynchronous response mechanism is used, a provisioning response confirming the request is being processed may be returned to the issuer before step 002. See details in semi-asynchronous clause 9.3.1.4.

9.3.2.23.3 Retrieve <*mgmtObj*>

The procedure is used to retrieve all or part information from an existing <*mgmtObj*> resource or its <*parameters*> sub-resource. Alternatively, the issuer can request to retrieve only a specific attribute or part of an attribute, as described in partial addressing, see clause 9.3.2.29.

Issuer: shall request to retrieve all or part of information from an existing <*mgmtObj*> resource by using a RETRIEVE verb. The request shall address a specific "<*mgmtObj*>" resource of an SCL or a specific <*parameters*> sub-resource of the <*mgmtObj*> resource to retrieve all attributes of the <*mgmtObj*> resource or the <*parameters*> sub-resource. Or the request shall address the individual attribute of the specific "<*mgmtObj*>" resource or the <*parameters*> sub-resource to retrieve the corresponding attribute value using the partial addressing mechanism as described in clause 9.3.2.29.

The issuer may be:

- An M2M NA.
- A D/GSCL on the remote entity.

Hosting SCL: shall check if the issuer has the READ permission on the addressed <*mgmtObj*> resource (or the <*parameters*> sub-resource in the case of sub-resource retrieval). Upon successful validation, the hosting SCL shall retrieve the corresponding <*mgmtObj*> resource or the <*parameters*> sub-resource including all attributes and references to all its child resources from its repository and shall respond to the issuer with the appropriate generic responses as indicated in clause 9.3.1.1. If the issuer is an M2M NA and if the requested information of the <*mgmtObj*> or <*parameters*> resource is not available on the hosting SCL, the hosting SCL shall identify the corresponding MO data on the remote entity according to the mapping relationship it maintains, and send the management request over mId reference point to get the corresponding MO data from the remote entity based on existing device management protocol (e.g. OMA-DM, BBF TR-069), then return the result to the issuer.

The hosting SCL shall be:

- An NSCL.

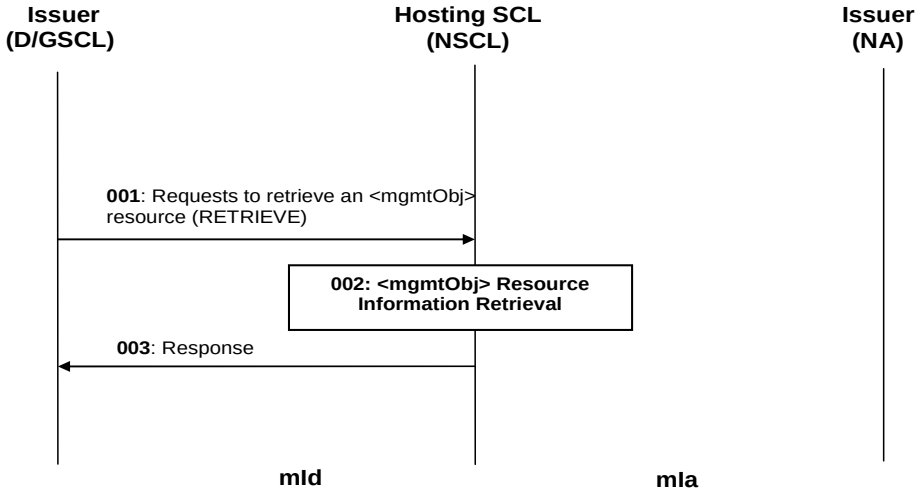


Figure 9.96: Illustration of procedures to retrieve an <*mgmtObj*> resource by a D/GSCL

Step 001: To retrieve an <*mgmtObj*> resource, D/GSCL shall issue a RETRIEVE request to the NSCL.

Step 002: If the retrieval is allowed, an <*mgmtObj*> resource shall be retrieved from the repository of NSCL.

Step 003: A response shall be returned to D/GSCL.

List of procedure specific exceptions:

Step 002: If the retrieval is not allowed or the specific <*mgmtObj*> resource does not exist in the NSCL, the retrieval shall be failed and a proper error code shall be returned to D/GSCL in step 003.

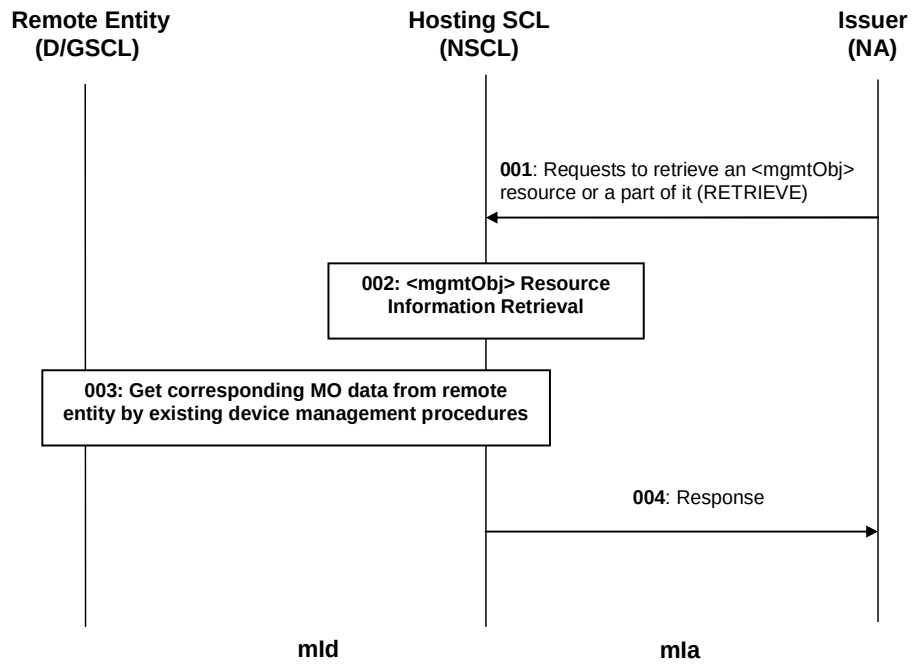


Figure 9.97: Illustration of procedures to retrieve an *<mgmtObj>* resource by an M2M NA

Step 001: To retrieve MO data from a remote entity (D/GSCL), NA shall issue a RETRIEVE request to the NSCL to retrieve the corresponding <*mgmtObj*> resource.

Step 002: If the retrieval is allowed, the addressed attributes of the <*mgmtObj*> resource or its sub-resource shall be retrieved from the repository of the NSCL.

Step 003 If the requested information is not available or expired in the <*mgmtObj*> resource, NSCL shall trigger existing (OMA-DM or BBF TR-069) device management procedures to get the latest information of the corresponding MO from the remote entity.

Step 004: A response shall be returned to NA.

List of procedure specific exceptions:

Step 002: If the retrieval is not allowed or the specific <*mgmtObj*> resource does not exist in the NSCL, step 003 shall be skipped and a proper error code shall be returned to NA in step 004.

Step 003: If the corresponding MO data cannot be retrieved from remote entity due to some reason (e.g. MO not found), a proper error code shall be returned to NA in step 004.

NOTE: In the case that asynchronous response mechanism is used, a provisioning response confirming the request is being processed may be returned to the issuer before step 003. See details in semi-asynchronous clause 9.3.1.4.

9.3.2.23.4 Update <mgmtObj>

The procedure is used to update all or part information of an existing <*mgmtObj*> resource or its <*parameters*> sub-resource. Alternatively, the issuer can request to update only a specific attribute or part of an attribute, as described in partial addressing, see clause 9.3.2.29.

Issuer: shall request to update all or part information of an existing <*mgmtObj*> resource by using a UPDATE, CREATE or DELETE verb. The request shall address:

- a specific <*mgmtObj*> resource or a specific <*parameters*> sub-resource of the <*mgmtObj*> resource on the hosting SCL to update the all attributes of the <*mgmtObj*> resource or the <*parameters*> sub-resource with new values when using UPDATE verb; or

- a specific <*mgmtObj*> resource or a specific <*parameters*> sub-resource of the <*mgmtObj*> resource on the hosting SCL to add an attribute to the <*mgmtObj*> resource or the <*parameters*> sub-resource when using CREATE verb; or
- the individual attribute of the specific <*mgmtObj*> resource or <*parameters*> sub-resource on the hosting SCL to override, add value to, or delete the corresponding attribute when using UPDATE, CREATE or DELETE verb according to the partial addressing mechanism as described in clause 9.3.2.29.

The issuer may be:

- An M2M NA.
- A D/GSCL on the remote entity.

Hosting SCL: shall check if the issuer has the WRITE permission on the address <*mgmtObj*> resource (or the <*parameters*> sub-resource in the case of sub-resource update). The hosting SCL shall also check the validity of provided attributes if any. Upon successful validation, the hosting SCL shall update, add or delete the corresponding attribute(s) of the <*mgmtObj*> resource or the <*parameters*> sub-resource accordingly. If the issuer is an M2M NA, the hosting SCL shall also identify the corresponding MO data on the remote entity according to the mapping relationship it maintains, and send the management request over mId reference point to update, add or delete the corresponding MO in the remote entity accordingly based on existing device management protocol (e.g. OMA-DM or BBF TR-069). The hosting SCL shall respond to the issuer with the appropriate generic responses as indicated in clause 9.3.1.1.

The hosting SCL shall be:

- An NSCL.

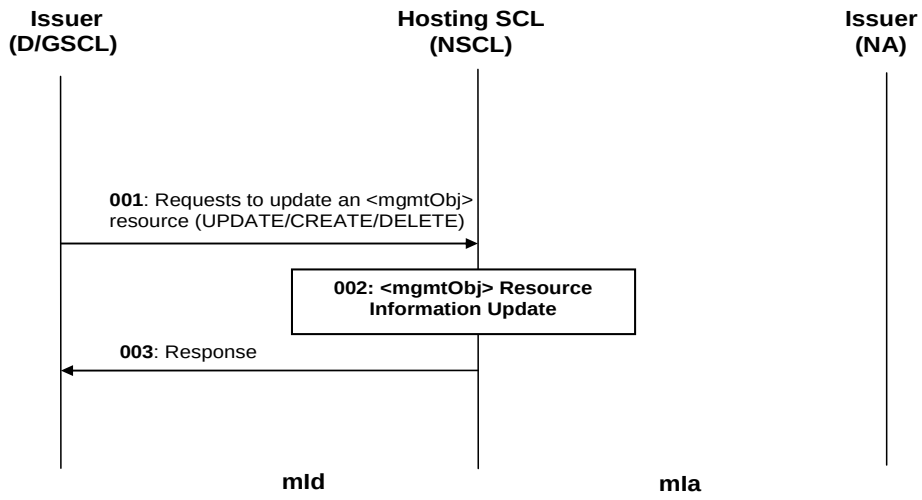


Figure 9.98: Illustration of procedures to update an <*mgmtObj*> resource by a D/GSCL

Step 001: To update an <*mgmtObj*> resource, D/GSCL shall issue a UPDATE/CREATE/DELETE request to the NSCL with the provided new value.

Step 002: If the update is allowed, an <*mgmtObj*> resource shall be updated from the repository of NSCL.

Step 003: A response shall be returned to D/GSCL.

List of procedure specific exceptions:

Step 002: If the update is not allowed or the specific <*mgmtObj*> resource does not exist in the NSCL, the deletion shall be failed and a proper error code shall be returned to D/GSCL in step 003.

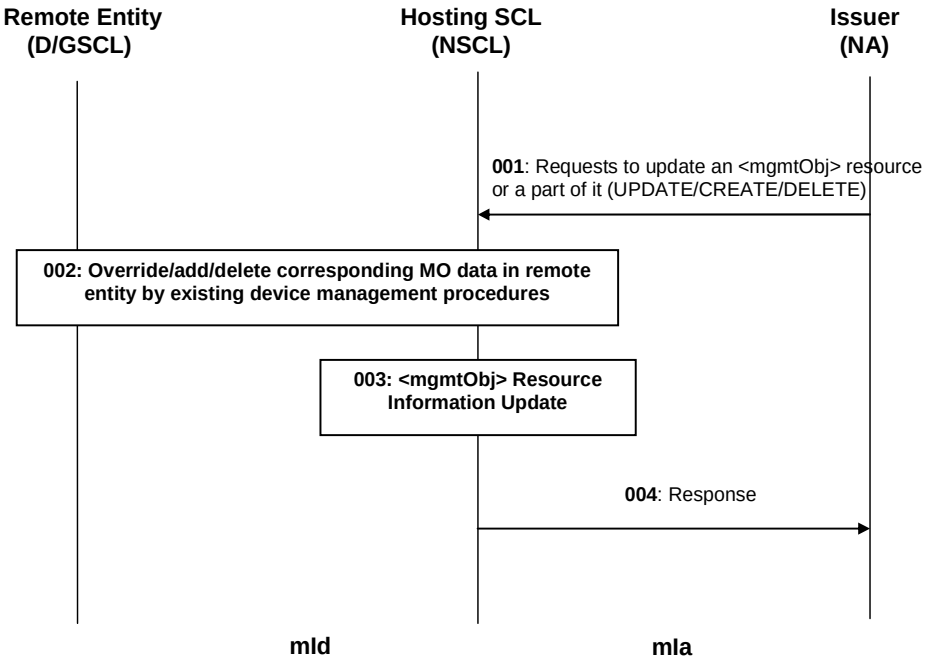


Figure 9.99: Illustration of procedures to update an <*mgmtObj*> resource or its sub-resource by an M2M NA

Step 001: To update MO data in a remote entity (D/GSCL), NA shall issue a UPDATE/ CREATE/ DELETE request to the NSCL to update the corresponding <*mgmtObj*> resource or its sub-resource.

Step 002: If the update is allowed, NSCL shall trigger existing (OMA-DM or BBF TR-069) device management procedures to override/ add/ delete the corresponding MO data in the remote entity accordingly.

Step 003: The addressed attribute(s) of the <*mgmtObj*> resource or its sub-resource shall be updated accordingly.

Step 004: A response shall be returned to NA.

List of procedure specific exceptions:

Step 002: If the update is not allowed or the specific <*mgmtObj*> resource (or its sub-resource) does not exist in the NSCL, steps 002-003 shall be skipped and a proper error code shall be returned to NA in step 004.

Step 002: If the corresponding MO data cannot be updated to remote entity due to some reason (e.g. not reachable, MO not found), step 003 shall be skipped and a proper error code shall be returned to NA in step 004.

Step 002: If the corresponding MO data cannot be deleted from remote entity due to some reason (e.g. not reachable, memory shortage or MO not found), step 003 shall still be performed and a successful response with the proper indication shall be returned to NA in step 004.

NOTE: In the case that asynchronous response mechanism is used, a provisioning response confirming the request is being processed may be returned to the issuer before step 002. See details in semi-asynchronous clause 9.3.1.4.

9.3.2.23.5 Delete <mgmtObj>

The procedure is used to delete an existing <*mgmtObj*> resource or its <*parameters*> sub-resource so as to hide the management function of a remote entity (i.e. M2M Device/Gateway) from being exposed via mIa reference point. An M2M NA uses this procedure to remove the corresponding MO data (e.g. an obsolete software package) from the remote entity.

Issuer: shall request to delete an existing <*mgmtObj*> resource or its <*parameters*> sub-resource by using a DELETE verb. The request shall address the specific <*mgmtObj*> resource or a specific <*parameters*> sub-resource of the <*mgmtObj*> resource on the hosting SCL.

The issuer may be:

- The D/GSCL on the remote entity: In this case, the D/GSCL issues the request to the hosting SCL to hide the corresponding remote management function from being exposed by the <*mgmtObj*> resource.
- An M2M NA: In this case, the NA requests the hosting SCL to delete the <*mgmtObj*> resource from the hosting SCL and to remove the corresponding MO data from the remote entity.

NOTE 1: The hosting SCL in the network domain may also delete the <*mgmtObj*> resource locally by itself. This internal procedure is out of scope.

NOTE 2: The <*mgmtObj*> resource may also be deleted in the hosting SCL by other offline provisioning means which are out of scope.

Hosting SCL: shall check if the issuer has the DELETE permission on the addressed <*mgmtObj*> resource (or the <*parameters*> sub-resource in the case of sub-resource deletion). Upon successful validation, the hosting SCL shall remove the addressed resource from its repository. If the issuer is an NA, the hosting SCL shall also identify the corresponding MO data on the remote entity according to the mapping relationship it maintains, and send the management request over mId reference point to remove the corresponding MO data from the remote entity based on existing management protocol. Then the SCL shall respond to the issuer with the appropriate generic responses as indicated in clause 9.3.1.1.

The hosting SCL shall be:

- An NSCL.

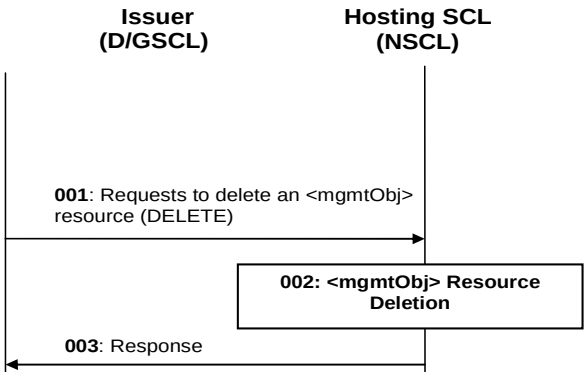


Figure 9.100: Illustration of procedures to delete an <*mgmtObj*> resource by D/GSCL

Step 001: To delete an <*mgmtObj*> resource, D/GSCL shall issue a DELETE request to the NSCL.

Step 002: If the deletion is allowed, an <*mgmtObj*> resource shall be removed from the repository of NSCL.

Step 003: A response shall be returned to D/GSCL.

List of procedure specific exceptions:

Step 002: If the deletion is not allowed or the specific <*mgmtObj*> resource does not exist in the NSCL, the deletion shall be failed and a proper error code shall be returned to D/GSCL in step 003.

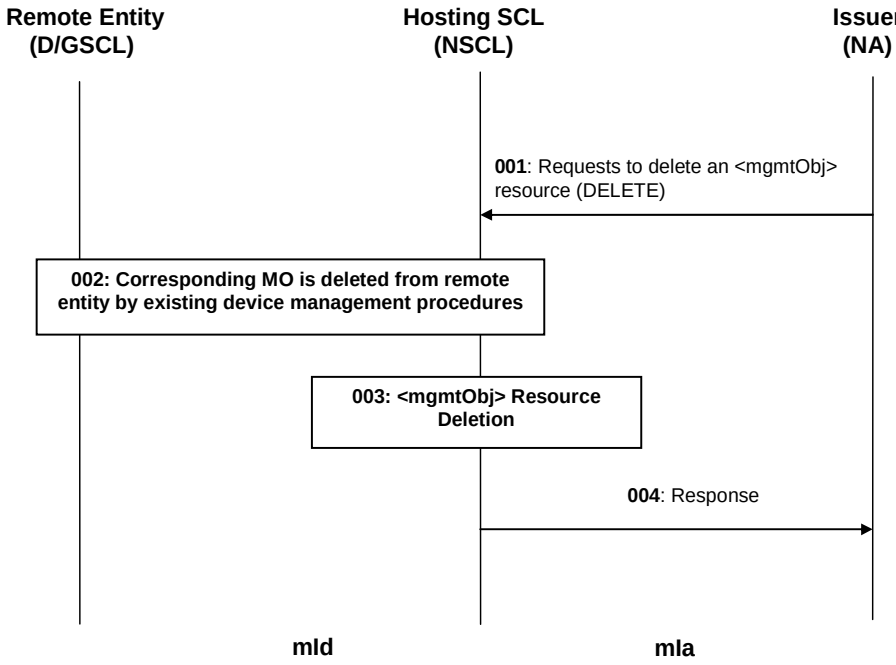


Figure 9.101: Illustration of procedures to delete an <*mgmtObj*> resource by NA

Step 001: To delete an existing MO from a remote entity (D/GSCL), NA shall issue a DELETE request to the NSCL to delete the corresponding <*mgmtObj*> resource.

Step 002: If the deletion is allowed, NSCL shall trigger existing (OMA-DM or BBF TR-069) device management procedures to delete the corresponding MO from the remote entity.

Step 003: The <*mgmtObj*> resource shall be deleted from the repository of the NSCL.

Step 004: A response shall be returned to NA.

List of procedure specific exceptions:

Step 002: If the deletion is not allowed or the specific <*mgmtObj*> resource does not exist in the NSCL, steps 002-003 shall be skipped and a proper error code shall be returned to NA in step 004.

Step002: If the corresponding MO cannot be deleted from remote entity due to some reason (e.g. not reachable, MO not found), step 003 shall still be performed and a successful response with the proper indication shall be returned to NA in step 004.

NOTE: In the case that asynchronous response mechanism is used, a provisioning response confirming the request is being processed may be returned to the issuer before step 002. See details in semi-asynchronous clause 9.3.1.4.

9.3.2.23.6 Execute <mgmtObj>

The procedure is used for executing a specific management command on a remote entity through an existing <*mgmtObj*> resource on the hosting SCL.

Issuer: shall request to execute a specific management command which is represented by an existing <*mgmtObj*> resource or its attribute/sub-resource by using an UPDATE verb. The request shall address the specific executable <*mgmtObj*> resource or its attribute/sub-resource and shall contain an empty body. In the case that the management command is represented by an attribute of a <*mgmtObj*> resource, such attribute shall contain a URI that the request may also address to trigger the command execution alternatively.

After the execution request, the issuer may request to retrieve the execution result or status from the executable <*mgmtObj*> resource or its attribute/sub-resource by using a RETRIEVE method as described in clause 9.3.2.23.3.

The issuer shall be:

- An M2M NA.

Hosting SCL: shall check if the issuer has the WRITE permission on the addressed <*mgmtObj*> resource or its attribute/sub-resource. Upon successful validation, the hosting SCL shall send the management request over mId reference point to execute the corresponding management command (e.g. "Exec" in OMA-DM [8])on the remote entity based on existing device management protocol. And the hosting SCL shall respond to the issuer with the appropriate generic responses as indicated in clause 9.3.1.1. The response may also contain execution results.

Upon receiving a RETRIEVE request to retrieve the execution result or status from the executable <*mgmtObj*> resource or its attribute/sub-resource, the hosting SCL shall perform the procedures as described in clause 9.3.2.23.3.

Upon receiving from remote entity a management notification (e.g. OMA-DM "Generic Alert" message [8] or BBF TR-069 "Inform" message [10]) regarding the execution result or status, the hosting SCL may actively send the management request over mId reference point to retrieve the execution result or status MO information from the remote entity and update the corresponding <*mgmtObj*> resource or its attribute/sub-resource locally.

The hosting SCL shall be:

- An NSCL.

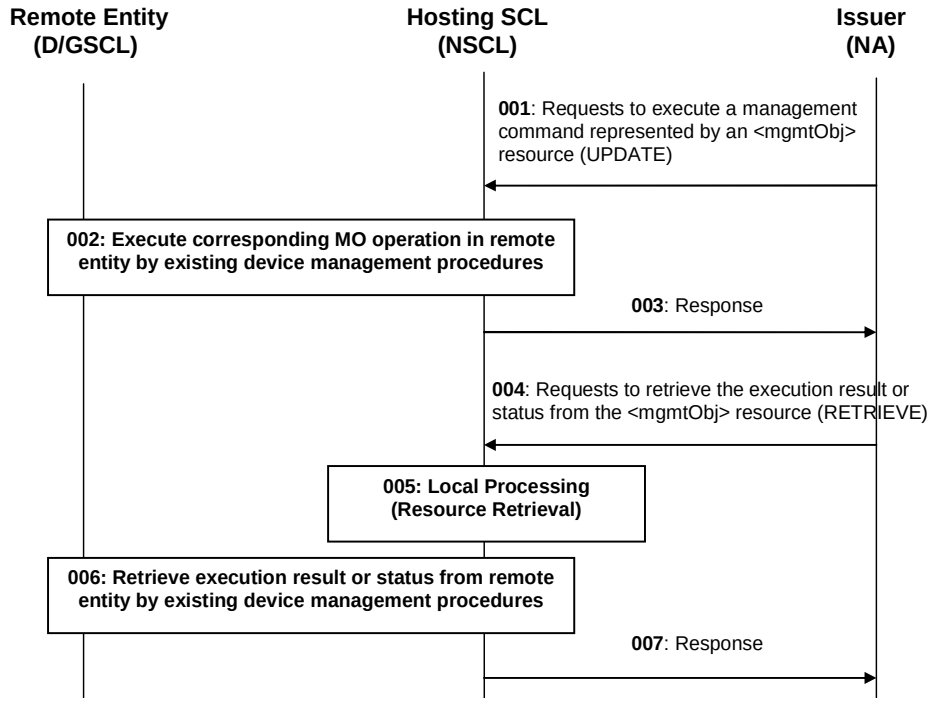


Figure 9.102: Illustration of procedures to execute an <*mgmtObj*> resource

Step 001: To execute a management command on a remote entity (D/GSCL), NA shall issue an UPDATE request to the NSCL to execute the corresponding executable <*mgmtObj*> resource or its attribute/sub-resource.

Step 002: If the execution is allowed, NSCL shall trigger existing (OMA-DM, BBF TR-069) device management procedures to execute the corresponding management command on the remote entity.

Step 003: A response shall be returned to NA.

Steps 004-007: NA issues RETRIEVE request to retrieve execution result or status from the executable <*mgmtObj*> resource or its attribute/sub-resource as described in clause 9.3.2.23.3.

List of procedure specific exceptions:

Step 002: If the execution is not allowed or the specified <*mgmtObj*> resource or its attribute/sub-resource does not exist in the NSCL, step 002 shall be skipped and a proper error code shall be returned to NA in step 003.

Step 002: If the corresponding MO operation cannot be executed in remote entity due to some reason (e.g. not reachable, MO not found), a proper error code shall be returned to NA in step 003.

NOTE: In the case that asynchronous response mechanism is used, a provisioning response confirming the request is being processed may be returned to the issuer before step 002. See details in semi-asynchronous clause 9.3.1.4.

9.3.2.23.7 Subscribe/Un-subscribe to <*mgmtObj*>

These procedures are used for subscribing for changes in an <*mgmtObj*> resource (or its <*parameters*> sub-resource) and managing the subscription itself. The procedures are described in detail in clause 9.3.2.19 with the following clarification.

- Before subscribing to an <*mgmtObj*> resource (or its <*parameters*> sub-resource), the subscriber may first need to issue an UPDATE request (as described in clause 9.3.2.23.4 or 9.3.2.23.6) to configure or execute the management function on the remote entity in order to generate notifications to the hosting SCL upon a specific management event.

- Upon receiving a management notification (e.g. OMA-DM "Generic Alert" message [8] or BBF TR-069 "Inform" message [10]) from the remote entity, the hosting SCL shall notify the subscriber as described in clause 9.3.2.19.6. If the updated MO information being subscribed to is not provided in the management notification, the hosting SCL shall first send a proper management request over mId reference point to retrieve required information based on existing management protocol, and update the corresponding <*mgmtObj*> (or its <*parameters*> sub-resource) resource locally before notifying the subscriber.

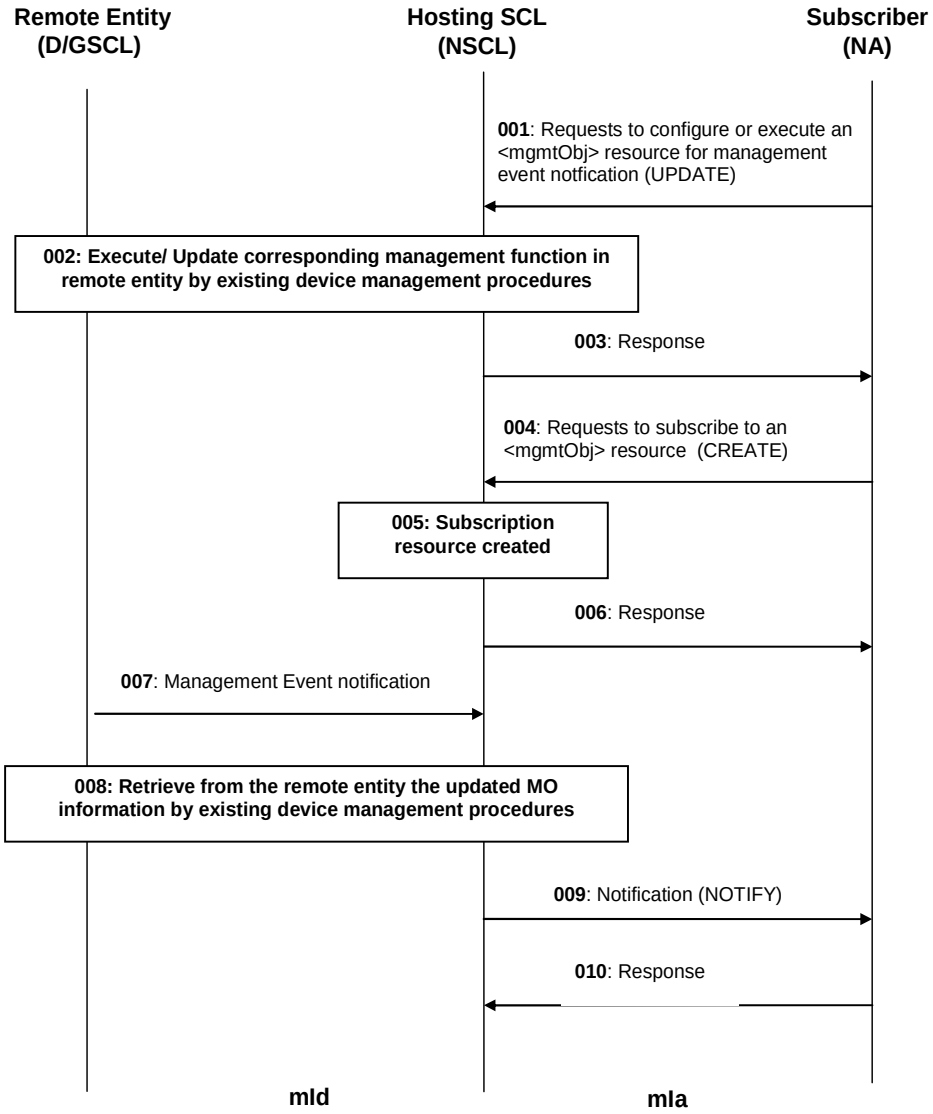


Figure 9.103: Illustration of procedures to subscribe/notify an <*mgmtObj*> resource

Steps 001-003: Optionally, NA issues an UPDATE request (as described in clause 9.3.2.23.4 or 9.3.2.23.6) to the NSCL to configure or execute an <*mgmtObj*> resource for management event notification.

Steps 004-006: NA issues a CREATE request (as described in clause 9.3.2.19.2) to subscribe to the changes of the <*mgmtObj*> resource.

Step 007: A management notification (e.g. OMA-DM "Generic Alert" message [8] or BBF TR-069 "Inform" message [10]) is sent from the remote entity to NSCL.

Step 008: Optionally, NSCL may retrieve the updated MO information based on existing management protocol, and update the corresponding <*mgmtObj*> resource locally.

Step 009: NSCL notify the subscriber as described in clause 9.3.2.19.6.

List of procedure specific exceptions:

Step 004: If the subscription is not allowed or the specified <*mgmtObj*> resource or its attribute/ sub-resource does not exist in the NSCL, step 005 shall be skipped and a proper error code shall be returned to NA in step 006. And the following steps will not be performed.

9.3.2.23.8 Manage a group of <*mgmtObj*> resources on different remote entities

In order to manage a group of <*mgmtObj*> resources (or <*parameters*> sub-resources) on different remote entities in a batch mode, a <*group*> resource shall be created in prior with the members as the list of target <*mgmtObj*> resources on different remote entities.

Issuer: sends a RESTful (CREATE/ RETRIEVE/ UPDATE/ DELETE) request to the group hosting SCL as described in clause 9.3.2.16, addressing the *membersContent* sub-resource of the <*group*> resource that represents the group of <*mgmtObj*> resources to be managed on different remote entities.

Group Hosting SCL: shall fan out the request to each of the member resource hosting SCL that hosts a <*mgmtObj*> resource of the group as described in clause 9.3.2.16.

Member Hosting SCL: shall perform the corresponding management procedures as described in clauses 9.3.2.23.1 to 9.3.2.23.7 based on the received RESTful request.

Note that the <mgmtObj> resources in the same group are recommended to be homogeneous, i.e. using the same data model and representing the same management function. Otherwise, the fan out requests may result in errors or meaningless response.

9.3.2.24 Management Command management

9.3.2.24.1 Introduction

This clause describes RESTful operations on an <*mgmtCmd*> resource over mIa and mId reference points. Different RESTful requests addressing an mgmtCmd resource (or its attributes or sub-resources) may be translated into existing management commands (such as those in BBF TR-069) and procedures performed on the remote entity, based on the converting mechanism described in the following clauses.

9.3.2.24.2 Create <mgmtCmd>

The procedure is used to create a specific <*mgmtCmd*> resource in the hosting SCL, which can be exposed at run-time over mIa reference point about the management commands supported in the remote entities (i.e. D/GSCL). The created <*mgmtCmd*> will be mapping from BBF TR-069 RPCs including FactoryReset, Reboot, Upload, Download, ScheduleDownload, ScheduleInform, CancelTransfer, ChangeDUState. (Other BBF TR-069 RPCs will be mapped to RESTful operations performed on associated <*mgmtObj*> resources).

Issuer: shall request- to create a new mgmtCmd type resource to be named as "<*mgmtCmd*>" by using a CREATE verb. The request shall address an "*mgmtObjs*" collection resource of the hosting SCL as described in clause 9.2.3.26. The request may also provide the attributes of the <*mgmtCmd*> to be created as described in clause 9.2.3.29 such as "cmdType".

The issuer may be:

- An M2M NA.
- The local SCL on the remote entity: In this case, the local SCL transforms supported management command into the <*mgmtCmd*> resource representation, then requests the hosting SCL to create the corresponding <*mgmtCmd*> resource.

NOTE 1: The hosting SCL in the network domain may also create the <*mgmtCmd*> resource locally by itself. The details are out of scope. Then an M2M NA can discover the created <*mgmtCmd*> and manipulate it.

NOTE 2: The <*mgmtCmd*> resource may also be created in the hosting SCL by other offline provisioning means which are out of scope.

Hosting SCL: shall check if the issuer has the CREATE permission on the addressed "*mgmtObjs*" resource. The hosting SCL shall also check the validity of provided attributes. Upon successful validation,, a new <*mgmtCmd*> resource with name "<*mgmtCmd*>" shall be created in the hosting SCL with the provided attributes. The hosting SCL shall maintain the mapping between the created <*mgmtCmd*> and corresponding RPCs in BBF TR-069 by the <cmdType> attribute of <*mgmtCmd*> resource. The hosting SCL shall respond to the issuer with the appropriate generic responses as indicated in clause 9.3.1.1. It shall also provide in the response the URL of the created new resource.

The hosting SCL shall be:

- An NSCL.

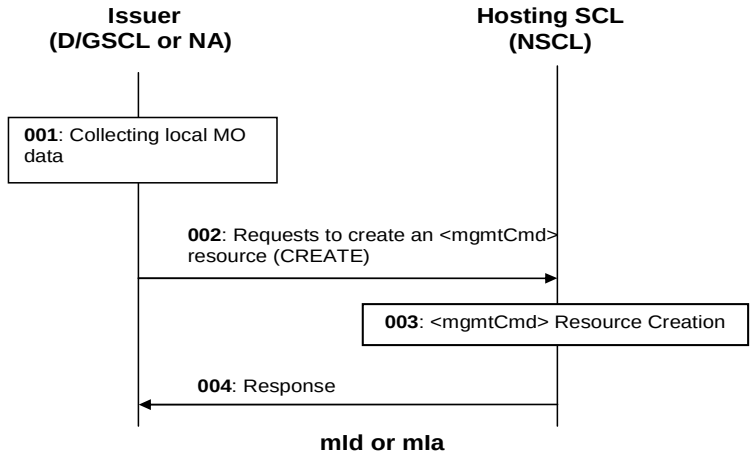


Figure 9.104: Illustration of procedures to create an <*mgmtCmd*> resource by D/GSCL

Step 001: This step is needed if and only if the issuer is D/GSCL. The D/GSCL shall first collect local management command to be exposed by the <*mgmtCmd*> resource.

Step 002: To create an <*mgmtCmd*> resource, the issuer shall issue a CREATE request with the collected MO data to the NSCL.

Step 003: If the creation is allowed, a <*mgmtCmd*> resource shall be created in the NSCL with the provided attributes.

Step 004: A response shall be returned to the issuer.

List of procedure specific exceptions:

Step 003: If the creation is not allowed, the <*mgmtCmd*> resource shall not be created, and a proper error code shall be returned to the issuer in step 004.

9.3.2.24.3 Retrieve <mgmtCmd>

The procedure is used to retrieve all or part information from an existing <*mgmtCmd*> resource. Alternatively, the issuer can request to retrieve only a specific attribute or part of an attribute, as described in partial addressing, see clause 9.3.2.29.

Issuer: shall request- to retrieve all or part of information from an existing <*mgmtCmd*> resource by using a RETRIEVE verb. The request shall address a specific "<*mgmtCmd*>" resource to retrieve all attributes and the references to the sub-resources of the <*mgmtCmd*> resource. Or the request shall address the individual attribute of the specific "<*mgmtCmd*>" resource to retrieve the corresponding attribute value using the partial addressing mechanism as described in clause 9.3.2.29.

The issuer may be:

- An M2M NA.

- A D/GSCL.

Hosting SCL: shall check if the issuer has the READ permission on the addressed <*mgmtCmd*> resource. Upon successful validation, the SCL shall retrieve the corresponding attributes the <*mgmtCmd*> resource and the references to the sub-resources from its repository and shall respond to the issuer with the appropriate generic responses as indicated in clause 9.3.1.1.

The hosting SCL shall be:

- An NSCL.

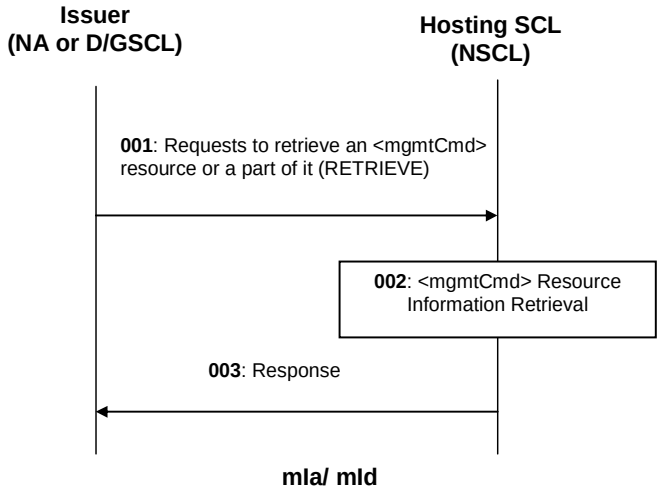


Figure 9.105: Illustration of procedures to retrieve an <*mgmtCmd*> resource

Step 001: NA (or D/GSCL) shall issue a RETRIEVE request to the NSCL to retrieve an <*mgmtCmd*> resource.

Step 002: If the retrieval is allowed, the addressed attributes of the <*mgmtCmd*> resource and the references to the sub-resources shall be retrieved from the repository of the NSCL.

Step 003 A response shall be returned to NA (or D/GSCL).

List of procedure specific exceptions:

Step 002: If the retrieval is not allowed or the specific <*mgmtCmd*> resource does not exist in the NSCL, a proper error code shall be returned to NA (or D/GSCL) in step 003.

9.3.2.24.4 Update <mgmtCmd>

The procedure is used to update all or part information of an existing <*mgmtCmd*> resource with new attributes. Alternatively, the issuer can request to update only a specific attribute or part of an attribute, as described in partial addressing, see clause 9.3.2.29.

Issuer: shall request to update all or part information of an existing <*mgmtCmd*> resource by using a UPDATE verb. The request shall address a specific "<*mgmtCmd*>" resource of an SCL to update all attributes of the <*mgmtCmd*> resource with the provided new value. Or the request shall address the individual attribute of the specific "<*mgmtCmd*>" resource to update the corresponding attribute with the provided new value using the partial addressing mechanism as described in clause 9.3.2.29.

The issuer may be:

- An M2M NA.
- A D/GSCL.

Hosting SCL: shall check if the issuer has the WRITE permission on the address <*mgmtCmd*> resource. The hosting SCL shall also check the validity of provided attributes if any. Upon successful validation, the hosting SCL shall overwrite the corresponding attributes of the <*mgmtCmd*> resource with the provided new data. The SCL shall respond to the issuer with the appropriate generic responses as indicated in clause 9.3.1.1.

The hosting SCL shall be:

- An NSCL.

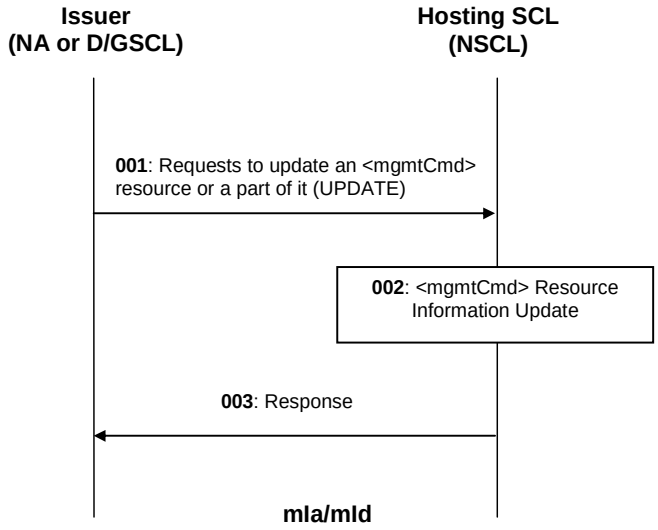


Figure 9.106: Illustration of procedures to update an <*mgmtCmd*> resource

Step 001: The NA (or D/GSCL) shall issue a UPDATE request to the NSCL to update the corresponding <*mgmtCmd*> resource.

Step 002: If the update is allowed, the addressed attributes of the <*mgmtCmd*> resource or its sub-resource shall be overridden with the provided new data.

Step 003: A response shall be returned to NA (or D/GSCL).

List of procedure specific exceptions:

Step 002: If the update is not allowed or the specific <*mgmtCmd*> resource (or its sub-resource) does not exist in the NSCL, steps 002 shall be skipped and a proper error code shall be returned to NA (or D/GSCL) in step 003.

9.3.2.24.5 Delete <mgmtCmd>

This procedure is used to delete an existing <*mgmtCmd*> resource. An M2M NA may also use this procedure to cancel all initiated <*execInstance*> of an <*mgmtCmd*> if applicable.

Issuer: shall request to delete an existing <*mgmtCmd*> resource by using a DELETE verb. The request shall address the specific "<*mgmtCmd*>" resource of a remote SCL on the hosting SCL.

The issuer may be:

- The D/GSCL on the remote entity: In this case, the local D/GSCL issues the request to the hosting SCL to hide the corresponding management command from being exposed by the <*mgmtCmd*> resource.
- An M2M NA: In this case, the NA requests the hosting SCL to delete the <*mgmtCmd*> resource from the hosting SCL and cancel all initiated <*execInstance*> of an <*mgmtCmd*> if applicable.

NOTE 1: The hosting SCL in the network domain may also delete an <*mgmtCmd*> resource locally by itself. This internal procedure is out of scope.

NOTE 2: The <*mgmtCmd*> resource may also be deleted in the hosting SCL by other offline provisioning means which are out of scope.

Hosting SCL: shall check if the issuer has the DELETE permission on the address <*mgmtCmd*> resource. Upon successful validation, the hosting SCL shall remove the resource from its repository. If the issuer is an NA and there is any initiated <*execInstance*> under the <*mgmtCmd*>that can be cancelled by a corresponding management command (e.g. CancelTransfer in BBF TR-069 [10]), the hosting SCL shall also issue the management command to the remote entity to cancel those initiated <*execInstance*> based on existing management protocol (i.e. BBF TR-069). Then the SCL shall respond to the issuer with the appropriate generic responses as indicated in clause 9.3.1.1.

The hosting SCL shall be:

- An NSCL.

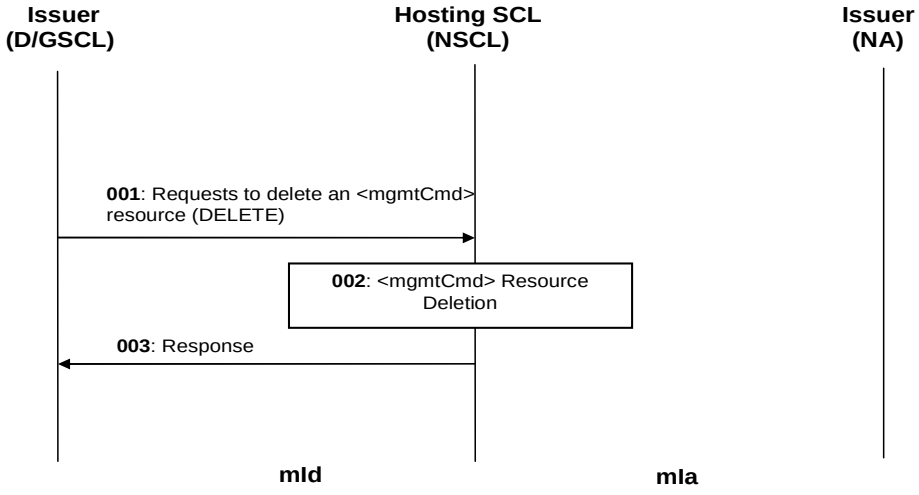


Figure 9.107: Illustration of procedures to delete an <*mgmtCmd*> resource by D/GSCL

Step 001: To delete an <*mgmtCmd*> resource, D/GSCL shall issue a DELETE request to the NSCL. Before this step, the D/GSCL may perform cancelling of the corresponding management command locally.

Step 002: If the deletion is allowed, the <*mgmtCmd*> resource shall be totally removed from the repository of NSCL.

Step 003: A response shall be returned to D/GSCL.

List of procedure specific exceptions:

Step 002: If the deletion is not allowed or the specific <*mgmtCmd*> resource does not exist in the NSCL, the deletion shall be failed and a proper error code shall be returned to D/GSCL in step 003.

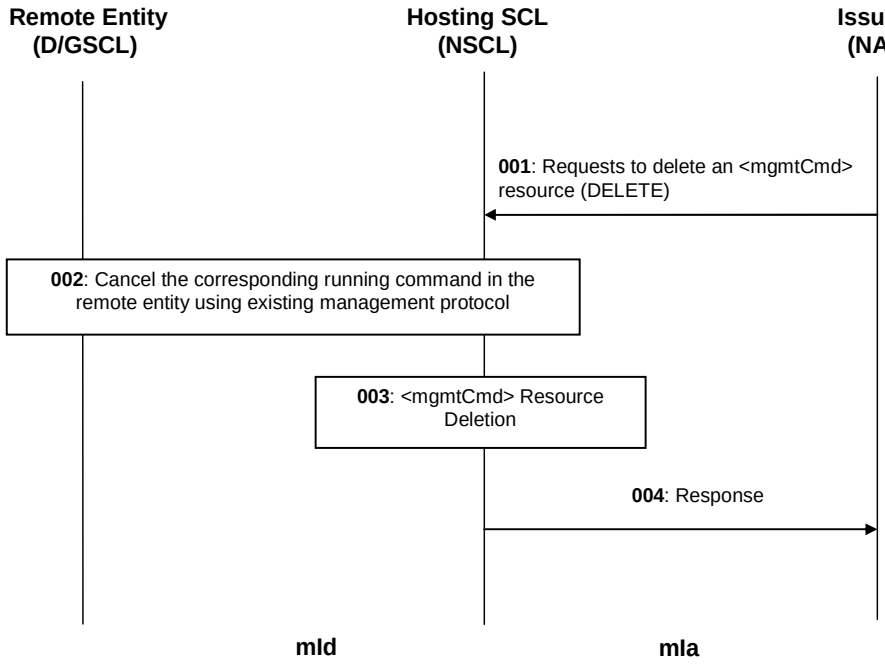


Figure 9.108: Illustration of procedures to delete an *<mgmtCmd>* resource by NA

Step 001: NA shall issue a DELETE request to the NSCL to delete an <*mgmtCmd*> resource.

Step 002: If there is any initiated <*execInstance*> under <*mgmtCmd*> and it is cancellable, NSCL shall cancel those initiated <*execInstance*> from the remote entity using corresponding management procedures in existing management protocol (such as CancelTransfer RPC in BBF TR-069).

Step 003: The <*mgmtCmd*> resource shall be deleted from the repository of the NSCL.

Step 004: A response shall be returned to NA.

List of procedure specific exceptions:

Step 002: If the deletion is not allowed or the specific <*mgmtCmd*> resource does not exist in the NSCL, steps 002-003 shall be skipped and a proper error code shall be returned to NA in step 004.

Step002: If the corresponding initiated commands cannot be deleted from remote entity due to some reason (e.g. not found), step 003 shall still be performed and a successful response with the proper indication shall be returned to NA in step 004.

NOTE 3: In the case that asynchronous response mechanism is used, a provisioning response confirming the request is being processed may be returned to the issuer before step 002. See details in semi-asynchronous clause 9.3.1.4.

9.3.2.24.6 Execute <mgmtCmd>

The procedure is used for executing a specific management command on a remote entity through an existing <*mgmtCmd*> resource on the hosting SCL.

Issuer: shall request to execute a specific management command which is represented by an existing <*mgmtCmd*> resource by using an UPDATE verb. The UPDATE request shall address the *execEnable* attribute of the <*mgmtCmd*> resource without any payload. Alternatively, the UPDATE request shall address the URI provided as the value of the *execEnable* attribute of the <*mgmtCmd*> resource.

After issuing the execution request, the issuer may request to retrieve the execution result or status from an <*execInstance*> sub-resource of the <*mgmtCmd*>by using a RETRIEVE method as described in clause 9.3.2.24.8.

The issuer shall be:

- An M2M NA.

Hosting SCL: shall check if the issuer has the WRITE permission on the addressed <*mgmtCmd*> resource. Upon successful validation, the hosting SCL shall perform command conversion and mapping, and send the converted management command over mId reference point to execute the corresponding management command with the provided arguments on the remote entity based on existing device management protocol (i.e. BBF TR-069). Then the hosting SCL shall create a corresponding <*execInstance*> resource under <*mgmtCmd*> for this command execution. And the hosting SCL shall respond to the issuer with the appropriate generic responses as indicated in clause 9.3.1.1. It shall also provide in the response the URL of the created <*execInstance*> resource.

Upon receiving from the remote entity a management notification (i.e. BBF TR-069 "Inform" message) regarding the execution result or status, the hosting SCL may actively send the management request over mId reference point to retrieve the execution result or status from the remote entity using existing management protocol (such as GetQueuedTransfers and GetAllQueuedTransfers RPCs in BBF TR-069), and update the corresponding <*execInstance*> sub-resource locally.

The hosting SCL shall be:

- An NSCL.

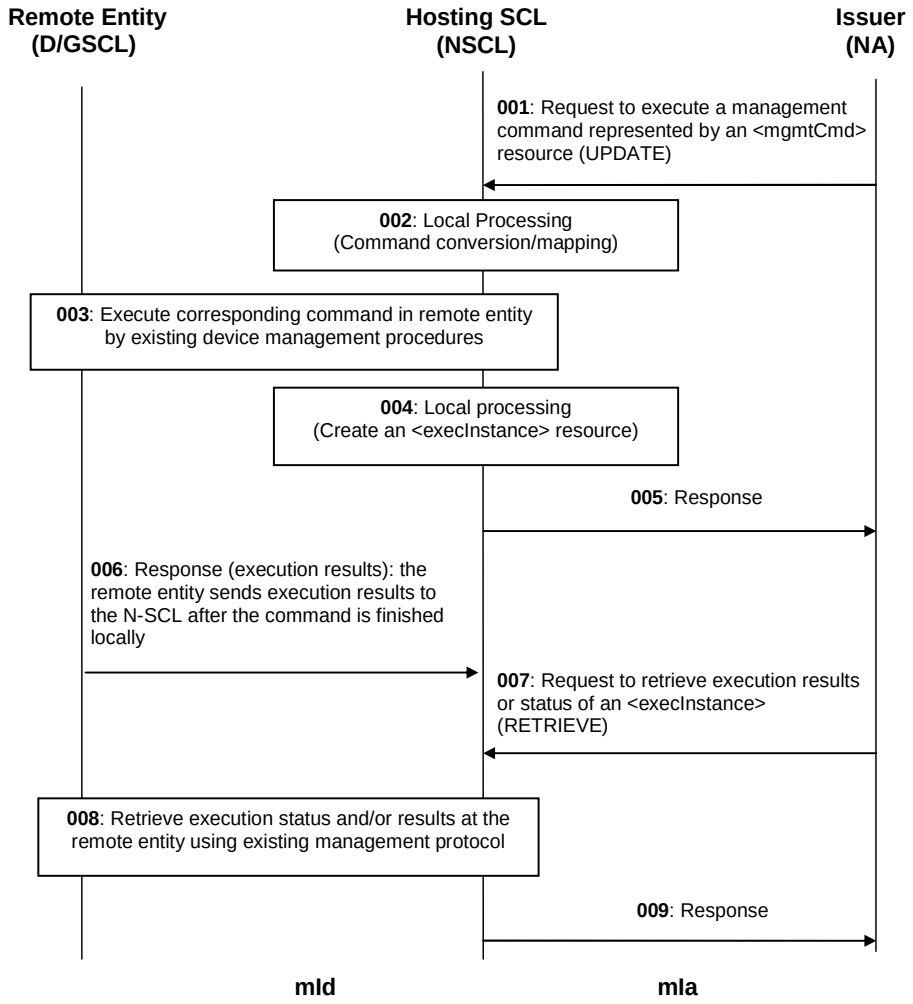


Figure 9.109: Illustration of procedures to execute a management command on a remote entity

Step 001: To execute a management command on a remote entity (D/GSCL), NA shall issue an UPDATE request to the NSCL to execute <*mgmtCmd*>. The UPDATE request shall address to "<*mgmtCmd*>/exec*Enable*" without any payload.

Step 002: If the execution is allowed, NSCL will do command conversion and mapping.

Step 003: The NSCL shall trigger existing device management procedures (i.e. BBF TR-069) to execute the corresponding management command on the remote entity.

Step 004: The NSCL shall perform local processing: if Step 3 is successful, the NSCL shall create an <*execInstance*> resource under the triggered <*mgmtCmd*> to maintain status and results for this execution.

Step 005: A response shall be returned to the NA.

The following steps are optional depending on the type of the command and execution status:

Step 006: After the command execution is finished, the remote entity sends response including execution results to the NSCL. The NSCL will store the execution results in corresponding <*execInstance*> resource. Note that this step can happen anytime after Step 5 such as after Step 007.

Step 007: The NA may use normal RETRIEVE procedure to retrieve the execution results or status of an <*execInstance*>.

Step 008: After receiving the RETRIEVE request from the NA, the NSCL can retrieve the execution status or results on the remote entity using existing management protocol. If step 006 occurs before step 007, step 008 may not be needed.

Step 009: A response shall be returned to the NA.

List of procedure specific exceptions:

Step 002: If the execution is not allowed or the specified <*mgmtCmd*> resource or its attribute/sub-resource does not exist in the NSCL, step 003 and step 004 shall be skipped and a proper error code shall be returned to NA in step 005.

Step 003: If the corresponding management command cannot be executed in remote entity due to some reason, step 004 shall be skipped and a proper error code shall be returned to NA in step 005.

9.3.2.24.7 Subscribe/Un-subscribe to <*mgmtCmd*>

9.3.2.24.8 Retrieve <execInstance>

The procedure is used to retrieve an existing <*execInstance*> resource including individual attributes. Alternatively, the issuer can request to retrieve only a specific attribute or part of an attribute, as described in partial addressing, see clause 9.3.2.29.

Issuer: shall request to retrieve all or part of information from an existing <*execInstance*> resource by using a RETRIEVE verb. The request shall address a specific "<*execInstance*>" resource of an hosting SCL to retrieve all attributes of the <*execInstance*> resource. Or the request shall address the individual attribute of the specific "<*execInstance*>" resource to retrieve the corresponding attribute value using the partial addressing mechanism as described in clause 9.3.2.29.

The issuer shall be:

- An M2M NA.

Hosting SCL: shall check if the issuer has the READ permission on the addressed <*mgmtCmd*> resource. Upon successful validation, the SCL shall retrieve the corresponding attributes of the <*execInstance*> resource and the references to the sub-resources from its repository and shall respond to the issuer with the appropriate generic responses as indicated in clause 9.3.1.1.

The hosting SCL may be:

- An NSCL.

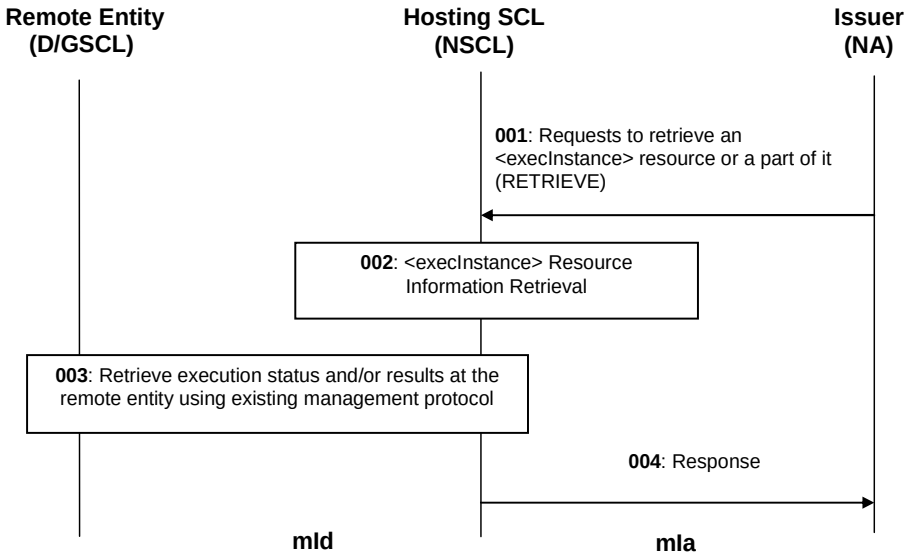


Figure 9.110: Illustration of procedures to retrieve an <*execInstance*> resource

Step 001: NA shall issue a RETRIEVE request to the NSCL to retrieve an <*execInstance*> resource.

Step 002: If the retrieval is allowed, the NSCL can retrieve the execution status or results on the remote entity using existing management protocol (i.e. BBF TR-069).

Step 003: If the retrieval is allowed, the addressed attributes of the <*execInstance*> resource shall be retrieved from the repository of the NSCL.

Step 004 A response shall be returned to NA.

List of procedure specific exceptions:

Step 002: If the retrieval is not allowed or the specific <*execInstance*> resource does not exist in the NSCL, Step 003 shall be skipped and a proper error code shall be returned to NA in Step 004.

9.3.2.24.9 Delete <execInstance>

The M2M Network Application uses this procedure to delete an existing <*execInstance*> resource.

Issuer: shall request to delete an existing <*execInstance*> resource by using a DELETE verb. The request shall address the specific "<*execInstance*>" resource of an <*mgmtCmd*> on the hosting SCL.

The issuer shall be:

- An M2M NA.

NOTE 1: The hosting SCL in the network domain may also delete an <*execInstance*> resource locally by itself. This internal procedure is out of scope.

NOTE 2: The <*execInstance*> resource may also be deleted in the hosting SCL by other offline provisioning means which are out of scope.

Hosting SCL: shall check if the issuer has the DELETE permission on the addressed <*execInstance*> resource. Upon successful validation, the hosting SCL shall remove the resource from its repository. If the <*execInstance*> is not finished on the remote entity and it is cancellable, the hosting SCL shall also use existing management protocol (i.e. BBF TR-069 CancelTransfer RPC) to cancel the corresponding management currently initiated at the remote entity. Then the SCL shall respond to the issuer with the appropriate generic responses as indicated in clause 9.3.1.1.

The hosting SCL shall be:

- An NSCL.

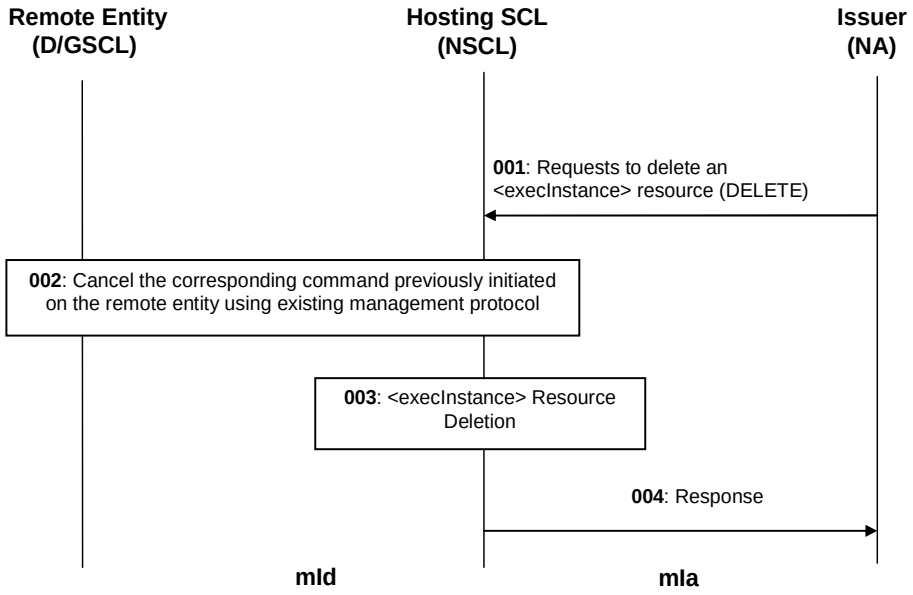


Figure 9.111: Illustration of procedures to delete an <*execInstance*> resource by an NA

Step 001: An NA shall issue a DELETE request to the NSCL to delete an <*execInstance*> resource.

Step 002: If the <*execInstance*> is not finished yet and it is cancellable, the NSCL shall cancel the <*execInstance*> from the remote entity using corresponding management procedures in existing management protocol (i.e. CancelTransfer RPC in BBF TR-069 [10]). If the <*execInstance*> is already complete or it is not cancellable, Step 002 shall be skipped.

Step 003: The <*execInstance*> resource shall be deleted from the repository of the NSCL.

Step 004: A response shall be returned to NA.

List of procedure specific exceptions:

Step 001: If the deletion is not allowed or the specific <*execInstance*> resource does not exist in the NSCL, steps 002-003 shall be skipped and a proper error code shall be returned to NA in step 004.

Step 002: If the corresponding initiated commands cannot be deleted from remote entity due to some reason (e.g. not found), step 003 shall still be performed and a successful response with the proper indication shall be returned to NA in step 004.

9.3.2.24.10 Cancel <execInstance>

The M2M NA uses this procedure to disable/stop/cancel an initiated management command execution on the remote entity through an existing <*execInstance*> resource on the hosting SCL.

Issuer: shall request to disable/stop/cancel an initiated management command execution which is represented by an existing <*execInstance*> resource by using an UPDATE verb. The UPDATE request shall address the *execDisable* attribute without any payload.

The issuer shall be:

- An M2M NA.

Hosting SCL: shall check if the issuer has the WRITE permission on the addressed <*mgmtCmd*> resource. Upon successful validation, the hosting SCL shall perform command conversion and mapping, then use existing management protocol (i.e. BBF TR-069) to cancel the corresponding management command execution initiated on the remote entity over mId reference point. And the hosting SCL shall respond to the issuer with the appropriate generic responses as indicated in clause 9.3.1.1. The response may also contain command cancellation results.

The hosting SCL shall be:

- An NSCL.

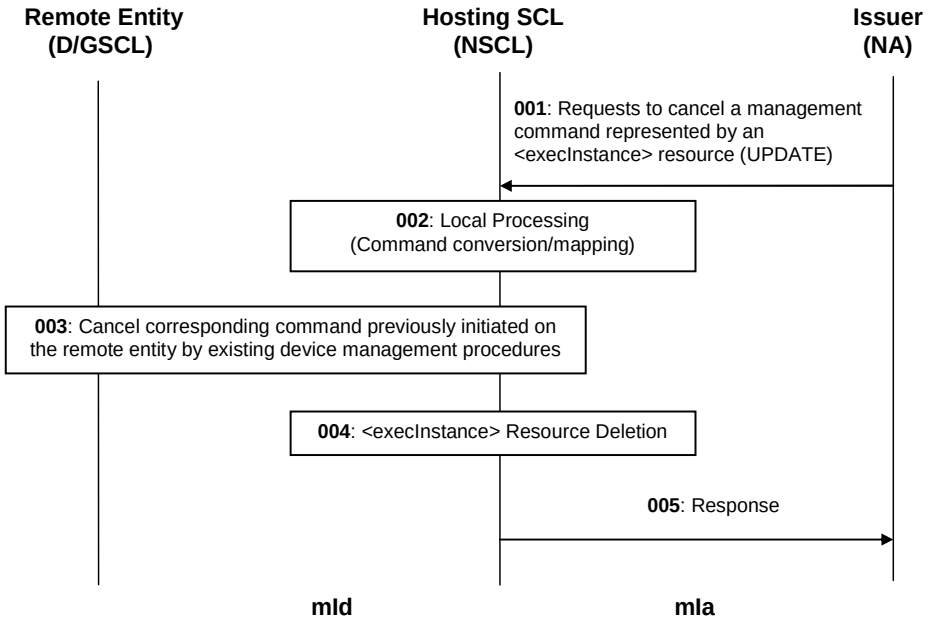


Figure 9.112: Illustration of procedures to cancel a management command execution instance on a remote entity

Step 001: To cancel a management command initiated on a remote entity (D/GSCL), an NA shall issue an UPDATE request to the NSCL to trigger to cancel a <*execInstance*>. The UPDATE request shall address to "<*execInstance*>/*execDisable*" without any payload.

Step 002: If the cancellation is allowed, NSCL will do command conversion and mapping.

Step 003: The NSCL shall cancel the <*execInstance*> from the remote entity using corresponding management procedures in existing management protocol (i.e. CancelTransfer RPC in BBF TR-069).

Step 004: The NSCL shall delete the corresponding <*execInstance*> resource.

Step 005: A response shall be returned to NA.

List of procedure specific exceptions:

Step 002: If the execution is not allowed or the specified <*execInstance*> resource does not exist in the NSCL or the <*execInstance*> is finished, step 003 and step 004 shall be skipped and a proper error code shall be returned to NA in step 003.

Step 003: If the corresponding management command cannot be executed in remote entity due to some reason, step 004 will still be performed.

9.3.2.25 Notification Channels Collection Management

9.3.2.25.1 Introduction

This clause describes different procedures for managing the collection resource *notificationChannels* as defined in clause 9.2.3.34. Such type of resource (like any other collection) cannot be created or deleted by means of a request over the reference points: mIa, mId and dIa.

9.3.2.25.2 Retrieve notificationChannels

This procedure is used for getting a list of references to all notification channels in the addressed collection resource and/or the attributes of the collection resource.

The procedure is described in details in clause 9.3.2.30.2.

9.3.2.25.3 Update notificationChannels

This operation is not applicable.

9.3.2.25.4 Subscribe/Un-Subscribe to *notificationChannels*

This operation is not applicable.

9.3.2.26 Notification Channel Management

9.3.2.26.1 Introduction

The procedures described below only apply for the case where the channelType is long-polling.

The <*notificationChannel*> resource offers a method for a client (SCL or application) that is not server capable to receive (semi-) asynchronous notifications the client has subscribed for. The method described here is based on blocking requests and often referred to as "long polling". The notifications are conveyed through a common notification channel and before a "long polling" request can be invoked a notification channel shall first be established.

The notification channel is created by creating a resource on the SCL which acts as a notification server. In addition to the URI of the created notification channel, the hosting SCL will provide two other URIs in the response. The issuer now uses this first URI as the contact URI when subscribing for notifications. A single notification channel may handle notifications from several resources and from several SCLs. Each SCL will send subsequent notifications using this contact URI pointing to the SCL acting as the notification server. In case the SCL where the subscription resides and the SCL acting as the notification server are the same entity, these notifications are internal.
The second URI is used to retrieve the notifications from the notification server using the "long-polling" mechanism. In the long polling mechanism a client does a long-polling request to which the server does not respond until a notification for that client becomes available. The server sends notifications as responses to this long polling request. The client immediately starts the next long-polling request to receive subsequent notifications. The problem with this mechanism is that a request cannot remain open indefinitely, due to network policies. Therefore, the server will also send an empty notification after a determined period. This will inform the client to do a new long polling request.

When the SCL acting as notification server performs the correlation between contactURI and long-polling URI for the notification channel, i.e. when it receives a notification from an SCL on a contactURI, it conveys the notification to the client with the response to the corresponding pending "long-polling" request.

A notification channel has certain time-to-live and will automatically be refreshed when accessed by a client by doing a "long-polling" request. The lifetime of a notification channel in case of inactivity is decided by SCL policy.

The design of the present document is done in such a way that it allows for the possibility to add support for other notification methods in addition to long-polling.

It should be noted that in order not to disclose underlying network topology, the SCL acting as notification server may send the client a mapped version of the real call-back address. In opposite direction, in such cases when the server receives such mapped URI, it will apply de-mapping of the URI before it can be used. How this mapping and de-mapping is performed on the server is out of scope for the present document.

9.3.2.26.2 Create <notificationChannel>

This procedure is used for creating a new notification channel for a client.

Issuer: shall request to create the <*notificationChannel*> resource using a CREATE verb. The request shall address the *notificationChannels* collection. The issuer provides a channelType of "long polling" and related channelData.
Although the specification does allow other types of channels to be established, only the long-polling type is defined at the moment.

Hosting SCL: shall validate the received request. Creation shall only be allowed if the issuer is allowed according to the default access rights for the *notificationChannels* resource. This effectively means that if the addressed resource is *<sclBase>/applications/<application>/notificationChannels* the issuer shall correspond to *<application>*. If the addressed resource is *<sclBase>/scls/<scl>/notificationChannels*, the issuer shall correspond to the *<scl>* resource.

The hosting SCL shall allocate a *contactURI*, a long-polling URI and then create a *<notificationChannel>* resource and gives it an expiration time according to server policies. The expiration time is not exposed to the client. The hosting SCL shall return a successful response that contains the representation of the newly created *<notificationChannel>* resource.

Issuer: shall initiate long polling request which shall address the returned *<notificationChannel>* resource. The long polling request is a RETRIEVE request with no contact.

Hosting SCL: shall validate the issuer of the long polling request. Only the original creator of the *<notificationChannel>* shall be allowed to perform the long polling request.

The issuer now may use the *contactURI* provided as part of the notification channel representation in any *<subscription>* resource. And it can use the long-polling URI for its long polling request. These two actions (subscribing and starting the long polling) can be done in any order, but it should be noted that the long-polling shall start before the notification channel expires.

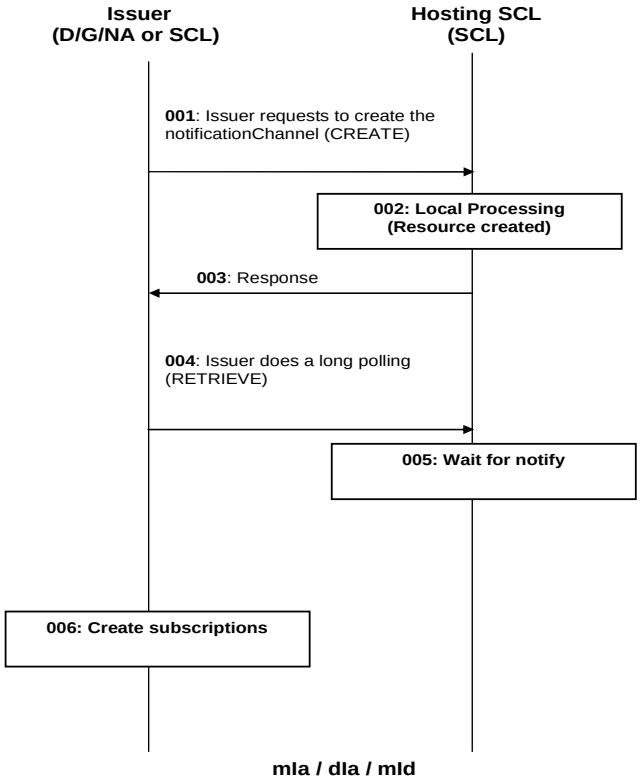


Figure 9.113: Procedures for *<notificationChannel>* create

Step 001: The Issuer requests to the Hosting SCL to create a *<notificationChannel>*.

Step 002: The Hosting SCL shall check if the Issuer is authorized to create the *<notificationChannel>* resource.

Step 003: The Hosting SCL responds positively to the request with the notification channel resource URI and the *<notificationChannel>* resource representation which includes the *contactURI* and the long polling URI.

Step 004: The Issuer initiates a long polling request with a long polling URI obtained in step 003 returned as part of the *<notificationChannel>*.

Step 005: The hosting SCL waits for notifications. These can either be internal (from the same SCL) or external (from another SCL).

Step 006: The Issuer creates a subscription using the contactURI obtained in step 003 according to procedures described in clause 9.2.3.23.

Step 006 can be repeated multiple times.

Steps 004-005 and the step 006 can be initiated in a different order.

List of main procedure specific exceptions:

Step 002: The Issuer is not authorized to create the resource. The Hosting SCL responds with an error.

Step 002: The request contains a not supported channelType. The Hosting SCL responds with an error.

Step 004: If the issuer waits too long with the long polling request the <*notificationChannel*> resource is removed by the hosting SCL. In such a case a subsequent request addressing the notification channel or the long polling URI will result in a "not found" error response.

9.3.2.26.3 Retrieve <notificationChannel>

This procedure is used to retrieve the attributes of a <*notificationChannel*> resource.

Issuer: shall issue a request to retrieve the information of a <*notificationChannel*> resource, using the RETRIEVE verb. The request shall address an <*notificationChannel*> resource of an SCL as defined in clause 9.2.3.35. The request can address the complete <*notificationChannel*> resource or it can address an individual attribute in the <*notificationChannel*> resource.

Hosting SCL: shall validate the received request. Retrieval shall only be allowed if the issuer is either the sclBase of the hosting SCL or the original creator of the <*notificationChannel*> resource. If the complete resource was addressed, the hosting SCL shall return a representation of the notificationChannel resource, with all the attributes defined in the <*notificationChannel*> resource. If an individual attribute is addressed, only the value of that attribute shall be returned. The hosting SCL shall return a generic response as indicated in clause 9.3.2.29.

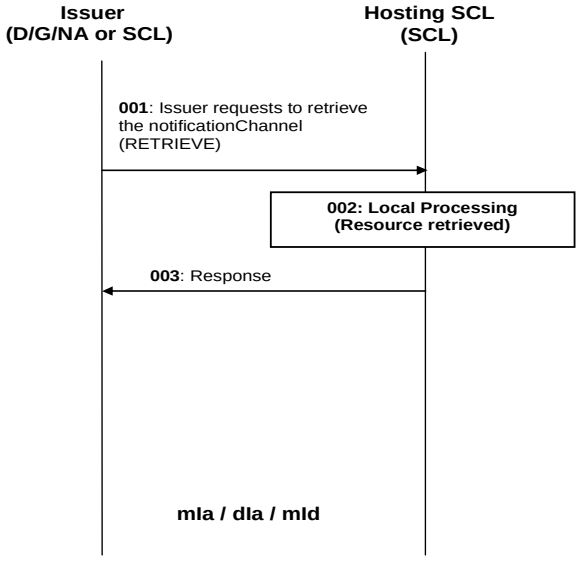


Figure 9.114: Procedures for <*notificationChannel*> retrieve

Step 001: The Issuer requests to the Hosting SCL to retrieve a <*notificationChannel*>.

Step 002: The Hosting SCL shall check if the Issuer is authorized to retrieve the <*notificationChannel*> resource.

Step 003: The Hosting SCL responds positively to the request with the representation of the resource.

List of main procedure specific exceptions:

Step 002: The Issuer is not authorized to retrieve the resource. The Hosting SCL responds with an error.

9.3.2.26.4 Update <notificationChannel>

This operation is not applicable.

9.3.2.26.5 Delete <notificationChannel>

This procedure is used to delete a specific <*notificationChannel*> resource.

Issuer: shall issue a request to delete a <*notificationChannel*> resource using the DELETE verb. The request addresses a <*notificationChannel*> resource of an SCL as defined in clause 9.2.3.35.

Hosting SCL: shall validate the received request. Delete shall only be allowed if the issuer is the <*sclBase*> or the original creator of the <*notificationChannel*> resource. The hosting SCL then shall delete the addressed <*notificationChannel*> resource. As a result, the long polling URI and the contact URI also shall become invalid. The hosting SCL shall return a generic response.

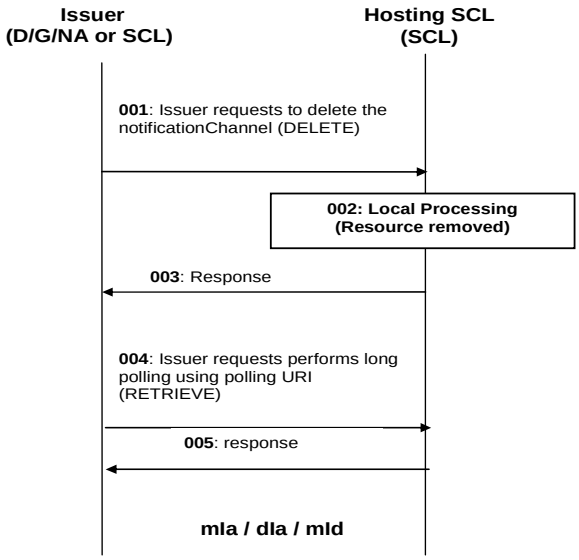


Figure 9.115: Procedures for <*notificationChannel*> delete

Step 001: The Issuer requests to the Hosting SCL to delete a <*notificationChannel*>.

Step 002: The Hosting SCL shall check if the Issuer is authorized to delete the <*notificationChannel*> resource.

Step 003: The Hosting SCL responds positively to the request.

Step 004: The issuer addresses the long polling URI provided as part of the deleted notification channel.

Step 005: The Hosting SCL sends an "not found" error response.

List of main procedure specific exceptions:

Step 002: The Issuer is not authorized to delete the resource. The Hosting SCL responds with an error.

9.3.2.26.6 Long polling based notifications delivered to issuer

This procedure is used to deliver notifications to the client using the notification channel resource. The pre-condition is that the application already created the <*notificationChannel*> resource (according to clause 9.3.2.26.2).

Issuer: shall issue a long polling request to the long-polling URI provided as a part of the notification channel creation. Note that this URI does not represent an actual resource that is represented in the resource structure. It does not have its own representation and does not have things like modification time or e-tags.

Hosting SCL: shall validate the issuer of the long polling request. Only the original creator of the <*notificationChannel*> is allowed to perform the long polling request. The Hosting SCL shall block the request until either a timeout occurs or until the notification is received.

The issuer shall subscribe to one of more resources using the subscription procedure defined in clause 9.2.3.19. The URI provided by the <*notificationChannel*> resource shall be used as a contact URI for the subscription. The SCL that hosts the subscription may or may not be the same as the SCL hosting the notification channel.

Note that the order of subscription and the initiation of long polling is not fixed, i.e. it is allowed to first issue a long polling request and then subscribe using the relevant contact URI. In fact, that order would provide less chance of missing the long polling deadline, since the notification channel will expire if not used.

Hosting Subscribed-to SCL: If a notification shall be sent (i.e. if a subscribed-to resource is modified), the subscribed-to SCL shall send a notification to the provided contact URI. If the subscribed-to SCL and the SCL hosting the notification channel (the notification server SCL) are different entities, this signalling is done using the mId REST API. If both are the same SCL, this signalling can be completely internally.

Hosting SCL: shall locate the notification channel resource corresponding to the contact URI. If a long polling is active for the long-polling URI corresponding to this notification channel, then the hosting SCL shall respond to the long polling request with a response that includes the notification data.

If no corresponding long polling is active when the notification is received, the notification itself is blocked for a short while, i.e. no response is returned to the subscribed-to SCL. The Issuer only has a limited time during which there is no active long polling, so if no long polling request is received in time, an error response is returned to the subscribed-to SCL.

Issuer: shall processes the response, and issue a new long polling request. The issuer has to initiate a new long polling request within a certain time.

There may be an enforced delay between the response and the next long polling request as well as a minimum time allowed for sending a new long polling request.

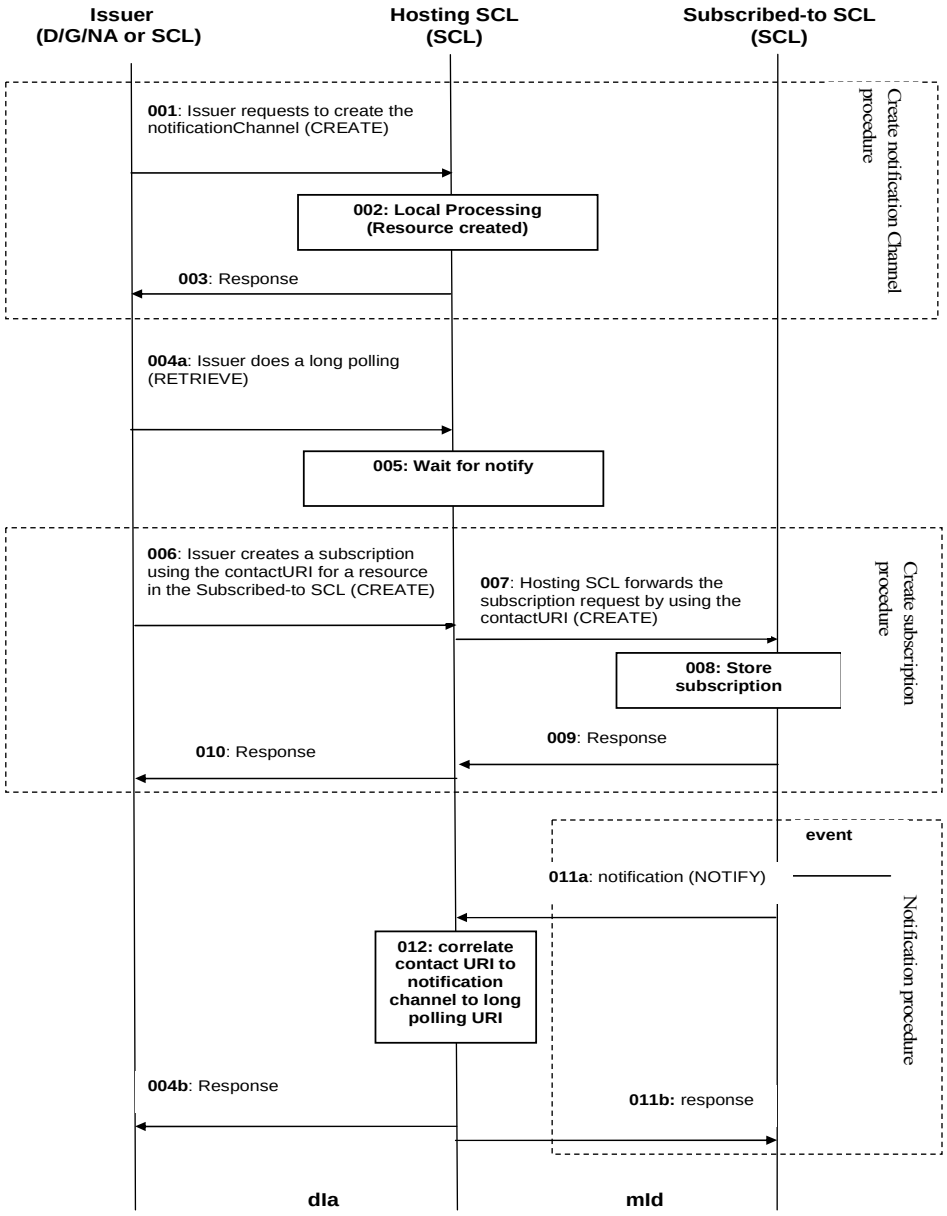


Figure 9.116: Procedures for long polling and notifications

Step 001-003: The issuer creates a <*notificationChannel*> resource and obtains a contact URI and a long polling URI, see clause 9.3.2.26.2.

Step 004a: The issuer performs a long polling request using the polling URI provided as part of the created notification channel.

Step 005: The hosting SCL blocks the request, i.e. does not send a response yet.

Step 006-010: The issuer creates a subscription using the subscription procedure as described in clause 9.2.3.19. The contact URI used in the subscription is the one obtained from an earlier created notification channel. The subscription can be created on the local SCL and co-located with the notification channel. Or the subscription can be created on a remote SCL.

Step 011a: After some time an event happens that triggers the subscribed-to SCL to send a notification using the contactURI obtained from the notification channel. If the subscribed-to SCL and the hosting SCL for the notification channel are the same, the notification is hosted internally otherwise it is transported over the mId.

Step 012: The hosting SCL correlates the contact URI to a notification channel and checks if there is a long polling request active on the long polling URI associated with that notification channel resource.

Step 011b: The hosting SCL sends a success response to the subscribed-to SCL.

Step 004b: The hosting SCL sends the notification data to the Issuer in the response to the long polling request.

List of main procedure specific exceptions:

Step 004: The long polling URI is no longer valid since the notification channel expired. A "not found" error response is returned and the procedure ends.

Step 005: No notification is received from the subscribed-to SCL within a certain time. The hosting SCL returns an empty response, and the sequence continues from step 010, i.e. the Issuer immediately issues a new long polling request.

Step 012: The contact URI does not correspond to an existing notification channel URI. In this case a "not found" result is returned to the subscribed-to SCL and the sequence continues from the previous step.

Step 012: No long polling request is active for the correlated notification channel. In this case the notification channel SCL waits for a limited time until a new long polling request is received and the flow continues from the next step. If the long polling request is not received within this time, an error response is returned to the subscribed-to SCL.

9.3.2.27 Resource Discovery

9.3.2.27.1 Introduction

The resource discovery procedures allow discovering of resources residing on an SCL. The use of filter criteria allows to limit the scope of the results. Filtering shall be performed on a subset of the offered resources' attributes using a query string. A match, that may include ranges, may be performed on the query string, and a successful response may be returned with a URI(s) list for resources that contains the matching attributes.

Resource discovery shall be done through the use of the following well known resourceURI: <*sclBase*>*/discovery* using the Retrieve method. The result shall be returned back to the issuer as part of the reply. The resource discovery procedures shall identify all matching resources from the entire hierarchy under <*sclBase*> (default behaviour). Optionally the discovery method may specify a prefix under which the discovery is performed. Other optional parameters may include the size of the answer (upper limit). The Hosting SCL may also implement a configured upper limit on the size of the answer. In such a case when both the issuer and the Hosting SCL have the upper limit, the upper limit in the Hosting SCL shall take precedence.

9.3.2.27.2 Resource discovery

This procedure shall be used for the discovery of resources under <*sclBase*> that match the provided filter criteria. The discovery result shall be returned to the issuer using a successful message.

Issuer: requests to discover (using Retrieve method) resources under a well known resource: <*sclBase*>*/discovery*. The issuer shall provide the <*sclBase*>*/discovery* well known URI in the request message. Filter criteria may be provided as a parameter by the issuer.

Hosting SCL: shall validate the received request and the validity of provided request parameters, The Hosting SCL shall respond to the issuer with the appropriate URIs list of discovered resources in the hosting SCL. If filter criteria is provided in the request, the Hosting SCL uses it identifying the resources whose attributes mach the filter criteria.

As a default behaviour the discovery procedure shall identify all matching resources from the entire hierarchy under <*sclBase*>. Optionally the Issuer may specify a prefix under which the discovery is performed.

If the size of the URI list is bigger than the Issuer requested size or the server configured size, then the full list is not returned. An incomplete list is returned and an indication is added in the response for warning the device. All the items of the returned list have to be valid resourceURIs, in particular the last one that has not to be truncated due to a too large size of the list.

The Hosting SCL shall not provide the actual content of the discovered resources.

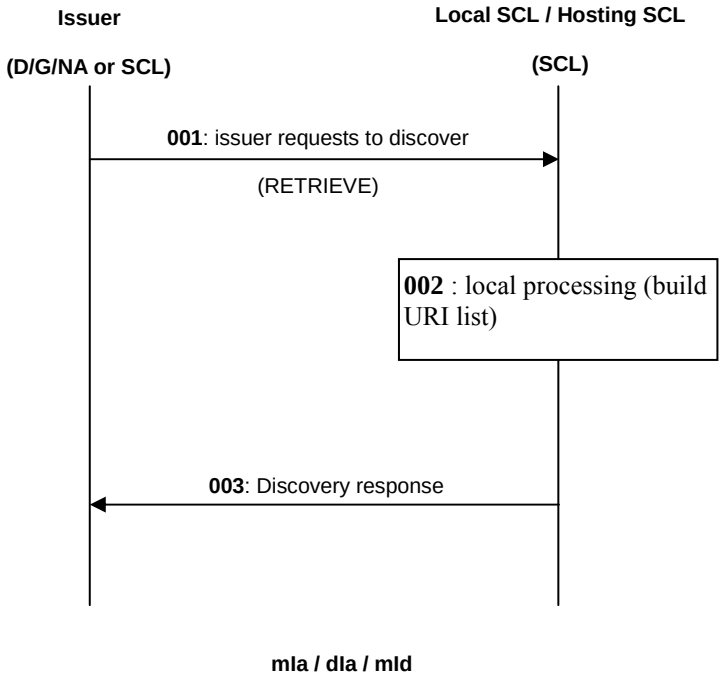


Figure 9.117: Discovery procedure message flow

Step 001: The issuer sends a request to the Hosting SCL for resource discovery using the Retrieve method.

The Issuer shall use *<sclBase>/discovery* well known URI in the request.

Step 002: The Hosting SCL checks the validity of the request. The Hosting SCL builds the URI list of the corresponding resources.

Step 003: The Hosting SCL returns a success Discovery response to the Issuer. The response contains the URI list of discovered resources.

List of main procedure specific exceptions:

Step 001: The request contains invalid parameters.

Step 001: The requesting Application or SCL is not registered.

9.3.2.28 Announce/De-Announce

This clause describes different procedures for announcing and de-announcing resources. It is assumed that prior to any of the procedures described below, the Issuer that performs the request has been properly Authenticated and Authorized.

9.3.2.28.1 Procedures to Announce Resource

9.3.2.28.1.1 Introduction

These procedures shall be used to announce a resource to all potential Announced-to SCLs, and are split between interaction on the mIa/dIa and interaction on the mId. An SCL may announce a resource only to SCLs to which it has registered and which are able to accept the announcement request.

The procedure to announce a resource may be triggered on the mIa/dIa reference point or on the mId. If the original issuer is an application then the announcement shall be triggered on the mIa/dIa and then executed on the mId. If the original issuer is an SCL then the announcement shall be triggered and executed on the mId. In the case, the original issuer is the SCL an application may subscribe to be notified of the status of the announcement.

9.3.2.28.1.2 Announce on dIa/mIa

Issuer: An application (DA, GA, NA) can request to announce a resource to other SCLs by changing the appropriate attribute of this resource.

The trigger of the announce procedure may be the registration of the Issuer to its Local SCL, the creation of a new resource(s) on the Local SCL, or an update of a resource(s) on the Local SCL.

The issuer may provide an announce attribute containing the type of announcement it is requesting, if any. The contents may include information regarding:

- The scope of the announcement (that is, if the announcement is to be made to specific SCLs, or if this decision is left to the local SCL).
- Whether the announce operation needs to be confirmed to the issuer.

a) Note that this should be added for all interface procedures. Mainly needed for CoAP binding.

- Announcement enabled (either ACTIVE or INACTIVE). This allows the attribute list to be populated, but without performing the announcement.

Inactive shall only be allowed at resource creation and shall be used to prevent the SCL to announce the resource, until activated. Once activated Inactive shall not be permissible.

b) The attribute list may be provided by an application:

- At registration and applicable to all announceable resources created by the issuer on its Local SCL. (Default attribute list):
  - Announceable resources may provide their own attribute list, in which case the default is not used.
  - Changes to the initial default attribute list are not propagated to previously created announceable resources. New announceable resources will use the updated default attribute list.
- For each resource created by the issuer on the Local SCL.

The issuer may modify the announce attribute list using an UPDATE to the resource.

Local SCL: The local SCL shall validate the received request. It shall trigger an Announce on mId procedure, only if the issuer is authorized to perform the received request according to the accessRights. The Local SCL returns a generic response according to clause 9.3.1.1.

9.3.2.28.1.3 Announce on mId

Hosting SCL: Based on the announce attribute list, the Hosting SCL announces a resource to the Announced-to SCL on mId using CREATE. The Hosting SCL forms the announced resource and sends an "announce resource" request to the Announced-to SCL, that includes the search strings and the link to the original resource. The Hosting SCL shall be able to announce the same resource to multiple SCLs. Unless specified in the announce attribute list, the Hosting SCL shall decide to which SCLs to announce to. Also, it is the responsibility of the Hosting SCL to provide the appropriate expiration time.

There are a few alternatives for the Response message, which will be sent depending on the issuers request as indicated in the bullets below:

- When the original issuer is an application and it does not indicate to which SCLs to announce to, the Hosting SCL sends a response to the original issuer (on the mIa/dIa) prior to the completion of the announced procedure (on the mId). The Hosting SCL decides when and to which SCLs to announce to (shown in Figure 9.118).
- When the original issuer is an application and it indicates to which SCLs to announce to, the Hosing SCL responds after it has completed announcing the resource to all indicated SCLs, thereby providing the issuer an indication of the status of the announced resources (shown in the Figure 9.119). The response shall provide a list of the SCLs that were announce to successfully.

- The announce request may indicate that no confirmation is required, therefore no response needs to be sent.

NOTE: Only in an UPDATE (on the mIa/dIa) of the AnnounceTo attribute without confirmation a no response is required. In any case that a CREATE is used (on the mIa/dIa) a response will always be generated. Currently on the mId a response is always required.

Announced-to SCL: The Announced-to SCL shall validate the received request and it shall create an announced resource with the specified attributes. Creation shall only be allowed if the Hosting SCL is authorized to create a child resource according to the accessRight defined. If the creation is successful, the Announced-to SCL shall return a successful response to the Hosting SCL, which includes an identifier (e.g. URI) of the created Announced Resource. If the creation is not successful the Announced-to SCL shall return an appropriate error message. Generic responses are indicated in clause 9.3.1.1.

Every "CREATE" on this Reference Point, mId, is for a single announce to an SCL.

In the case that the issuer may want to know to which SCLs the resource is announced to, it can subscribe.

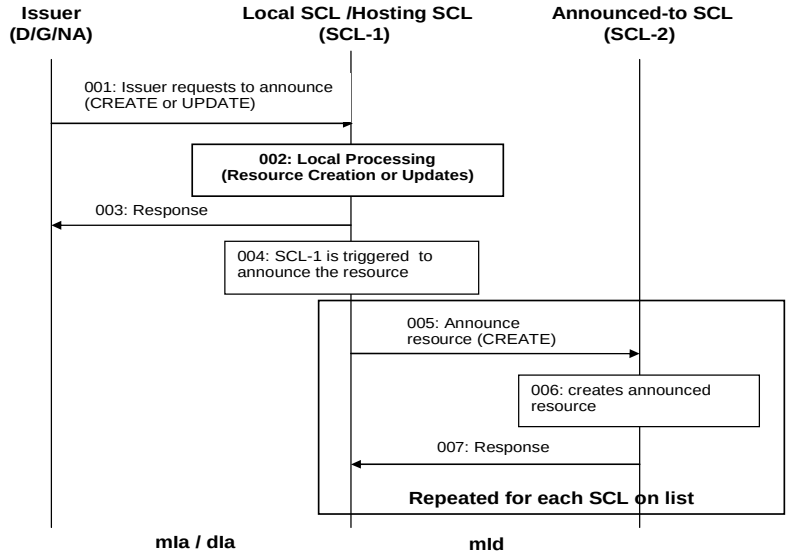


Figure 9.118: Illustration of Procedures to Announce Resources w/ immediate Response

Step 001: To announce a resource, the issuer shall issue a CREATE or UPDATE request. The issuer is an application and the receiver is the local SCL (Local SCL may also be the Hosting SCL). If the trigger of the announcement is due to the registration of an application or the creation of a new resource a CREATE request is issued. If the trigger of the announcement is due to an update of a resource an UPDATE request is issued.

Step 002: If the announce request is allowed by the Local or Hosting SCL, the resource shall be created or updated. If no characteristics are provided by the issuer, the resource shall be created with default characteristics determined by the addressed SCL.

Step 003: A generic result shall be returned as a response to the issuer, confirming that the issuer is authorized to perform the received request. The Local/Hosting SCL sends this response prior to the completion of the announce procedure and decides when and to which SCLs to announce to.

Step 004: The Local/Hosting SCL shall trigger an Announce procedure on the mId.

Step 005: The Local/Hosting SCL forms the announced resource and sends a CREATE request to the Announced-to SCL on the mId. The same resource may be announced to multiple SCLs using the same procedure.

Step 006: The Announced-to SCL shall validate the received request and then create an announced resource with the specified attributes if the Local/Hosting SCL is authorized.

Step 007: The Announced-to SCL shall return a response indicating if the creation was successful or unsuccessful.

Steps 005-007: Repeated for each SCL on the list to be announced to. The Announced-to list may be provided by the issuer (NA / DA/ GA) or decided by the Local/Hosting SCL.

List of main procedure specific exceptions:

Step 002: The requesting application is not registered. Steps 004 - 007 are not performed.

Step 002: The requesting application does not have the authorization to create this resource. Steps 004 - 007 are not performed.

Step 002: The provided characteristics are not acceptable to the Local / Hosting SCL. Steps 004 - 007 are not performed.

Step 006: The requesting SCL is not registered.

Step 006: The requesting SCL does not have the authorization to create this resource.

Step 006: The provided characteristics are not acceptable to the Announced-to SCL.

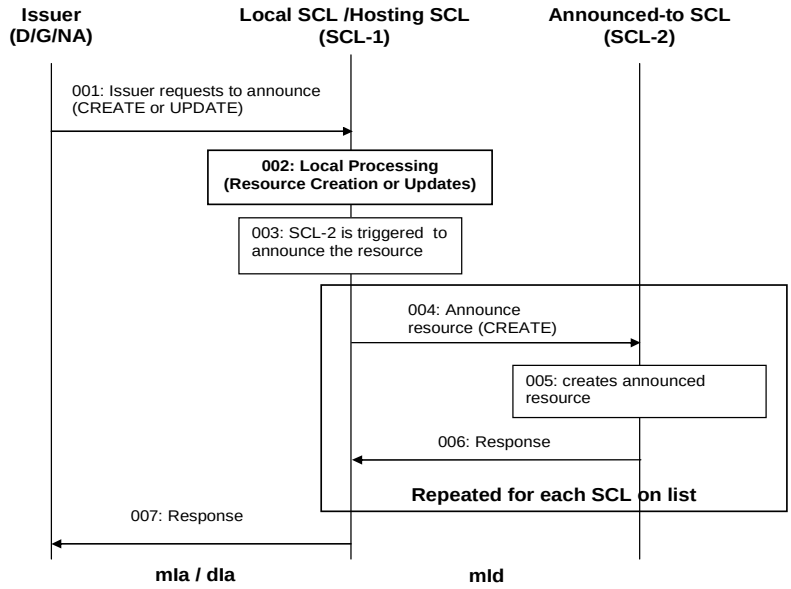


Figure 9.119: Illustration of Procedures to Announce Resources w/ delayed Response

Step 001: To announce a resource, the issuer shall issue a CREATE or UPDATE request. The issuer is an application and the receiver is the local SCL (Local SCL may also be the Hosting SCL). If the trigger of the announcement is due to the registration of an application or the creation of a new resource a CREATE request is issued. If the trigger of the announcement is due to an update of a resource an UPDATE request is issued.

Step 002: If the announce request is allowed by the Local or Hosting SCL, the resource shall be created or updated. If no characteristics are provided by the issuer, the resource shall be created with default characteristics determined by the addressed SCL.

Step 003: The Local / Hosting SCL shall trigger an Announce procedure on the mId.

Step 004: The Local / Hosting SCL forms the announced resource and sends a CREATE request to the Announced-to SCL on the mId. The same resource may be announced to multiple SCLs using the same procedure.

Step 005: The Announced-to SCL shall validate the received request and then create an announced resource with the specified attributes if the Local / Hosting SCL is authorized.

Step 006: The Announced-to SCL shall return a response indicating if the creation was successful or unsuccessful.

Steps 004-006: Repeated for each SCL on the list to be announced to. The Announced-to list may be provided by the issuer (NA/DA/GA) or decided by the Local/Hosting SCL.

Step 007: The Local/Hosting SCL sends this response after completion of the announce procedure and provides a list of the SCLs that were announced to successfully.

List of main procedure specific exceptions:

Step 002: The requesting application is not registered. Steps 003 - 006 are not performed and Step 007 is performed.

Step 002: The requesting application does not have the authorization to create this resource. Steps 003 - 006 are not performed and Step 007 is performed.

Step 002: The provided characteristics are not acceptable to the Local/Hosting SCL. Steps 003- 006 are not performed and Step 007 is performed.

Step 005: The requesting SCL is not registered.

Step 005: The requesting SCL does not have the authorization to create this resource.

Step 005: The provided characteristics are not acceptable to the Announced-to SCL.

9.3.2.28.2 Procedures to Update Announced Resources

This procedure is used to update a previously announced resource. The update may occur only on the mId Reference Point, or it may involve the dIa/mIa Reference Points when any change in the original resource results in a change of the announced resource.

9.3.2.28.2.1 Update Announced Resources on dIa/mIa

When an application requests changes in its created resource on the Local SCL (Hosting SCL), this may trigger the Hosting SCL to update of an announced resource. This update is only necessary when the searchStrings, accessRightID of the original resource changes. For example, there may be a content change in the original resource, and this can cause the change of searchStrings in an announced resource.

Refer to clause 9.3.2.8 Application Management for the detailed procedures on change of application resources.

9.3.2.28.2.2 Update Announced Resources on mId

Issuer (Hosting SCL): composes a request message to update an announced resource using the UPDATE verb on the mId Reference Point. Only the Hosting SCL which announced the resource previously can update the announced resource.

Announced-to SCL: updates the announced resource according to the request. If the update of the announced resource succeeded, it shall return the success code to the Issuer (Hosting SCL). Otherwise, it shall return an error code.

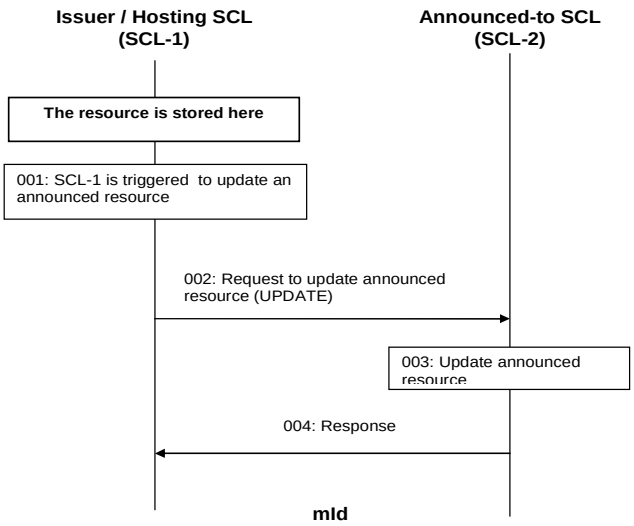


Figure 9.120: Illustration of Procedures to Update an Announced Resource

Step 001: A trigger to update an announce resource is generated due to a change in either the searchStrings or accessRightID of the original resource.

Step 002: The Hosting SCL forms the announced resource and sends an UPDATE request to the Announced-to SCL on the mId.

Step 003: The Announced-to SCL shall validate the received request and then update the announced resource with the specified attributes if the Hosting SCL is authorized.

Step 004: The Announced-to SCL shall return a response indicating if the update was successful or unsuccessful.

List of main procedure specific exceptions:

Step 003: The requesting SCL is not registered.

Step 003: The requesting SCL does not have the authorization to update this resource.

Step 003: The provided characteristics are not acceptable to the Announced-to SCL.

9.3.2.28.3 Procedures to De-Announce Resources

These procedures are used to de-announce a previously announced resource. De-announce can be implicit if the Announced Resource has an expiration time. De-announcing a previously announced resource shall not have any impact on the original resource.

9.3.2.28.3.1 De-Announce on dIa/mIa

Issuer: An application (DA,GA, NA) may request to de-announce previously announced resources. The triggers of de-announce could be that the Issuer performed an update or delete to the original resource.

Local SCL: The Local SCL shall validate the received request. It shall trigger a De-Announce on mId procedure, only if the issuer is authorized to perform the received request according to the accessRights. The Local SCL returns a generic response to the Issuer according to clause 9.3.1.1.

The Local SCL may respond to the De-Announce request on the dIa/mIa without waiting for the results from the mId Reference Point. If De-Announce is triggered by the deletion of the original resource, the Local SCL shall respond immediately. Alternatively, the Hosting SCL may respond after it has completed de-announcing the resource on mId, thereby providing the issuer an indication as to the status of the de-announced resources (shown in the Figure 9.121).

9.3.2.28.3.2 De-Announce on mId

Hosting SCL: The Hosting SCL may decide to De-Announce a resource as a result of an expiration of the announcement, or based on triggers from the Issuer (change of announce attribute list). The Hosting SCL sends a "De-Announce" request message using DELETE, over the mId Reference Point, to the Announced-to SCL to de-announce a previously announced resource. In the De-Announce request, the Hosting SCL may indicate one or multiple announced resources. It is the responsibility of the Hosting SCL to delete all announced resources.

Announced-to SCL: The Announced-to SCL shall validate the received request and it shall delete the announced resource(s) with the specified attributes. Deletion shall only be allowed if the Hosting SCL is authorized to delete a child resource according to the accessRight defined. If the deletion is successful, the Announced-to SCL shall return a successful response to the Hosting SCL. If the deletion is not successful the Announced-to SCL shall return an appropriate error message. Generic responses are indicated in clause 9.3.1.1.

De-Announce can be implicit if the Announced Resource has an expiration time. A response message may be sent to the Hosting SCL (announcing SCL) to indicate that the announced resource is de-announced. In cases when the Hosting SCL (announcing SCL) is not server capable, the subscription mechanism can be used.

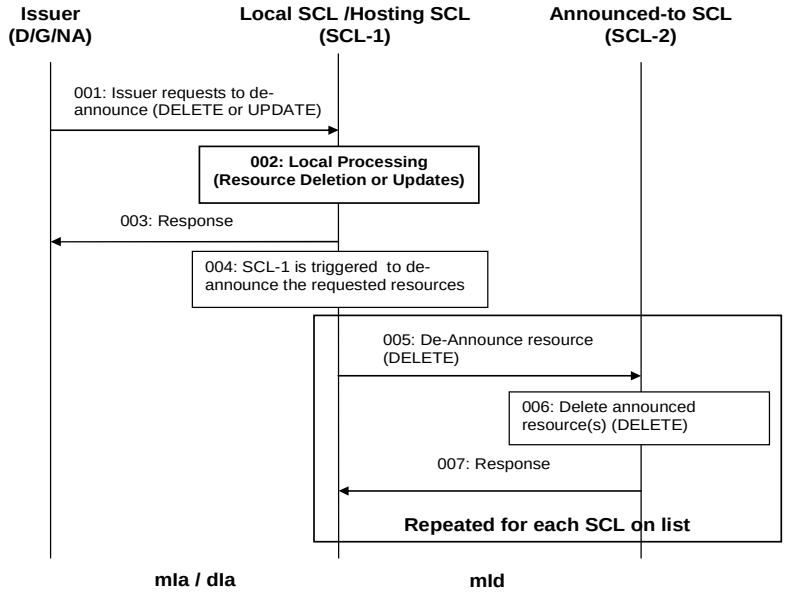


Figure 9.121: Illustration of Procedures to De-Announce Resources

Step 001: To de-announce a resource, the issuer shall issue a DELETE or UPDATE request. The issuer is an application and the receiver is the local SCL (Local SCL may also be the Hosting SCL). If the trigger of the de-announcement is due to the deletion of a resource a DELETE request is issued. If the trigger of the de-announcement is due to an update of a resource an UPDATE request is issued.

Step 002: If the de-announce request is allowed by the Local or Hosting SCL, the resource shall be deleted or updated.

Step 003: A generic result shall be returned as a response to the issuer, confirming that the issuer is authorized to perform the received request. The Local / Hosting SCL sends this response prior to the completion of the de-announce procedure and decides when to perform the de-announce procedure on the mId.

Step 004: The Local/Hosting SCL shall trigger a De-Announce procedure on the mId.

Step 005: The Local/Hosting SCL sends a DELETE request to the Announced-to SCL on the mId. The same resource may be de-announced to multiple SCLs using the same procedure.

Step 006: The Announced-to SCL shall validate the received request and then delete the announced resource if the Local/Hosting SCL is authorized.

Step 007: The Announced-to SCL shall return a response indicating if the deletion was successful or unsuccessful.

Steps 005-007: Repeated for each SCL on the list to be de-announced to. The De-Announced-to list may be provided by the issuer (NA/DA/GA) or decided by the Local/Hosting SCL.

List of main procedure specific exceptions:

Step 002: The requesting application is not registered. Steps 004 - 007 are not performed.

Step 002: The requesting application does not have the authorization to delete or update this resource. Steps 004 - 007 are not performed.

Step 002: The provided characteristics are not acceptable to the Local/Hosting SCL. Steps 004 - 007 are not performed.

Step 006: The requesting SCL is not registered.

Step 006: The requesting SCL does not have the authorization to delete this resource.

9.3.2.29 Partial addressing

9.3.2.29.1 Introduction

In REST a resource is normally manipulated in its entirety. I.e. the issuer provides a new representation that replaces the old representation. However, REST does not define a generic, fully specified and widely supported mechanism for modifying the parts of the resource representation in an idempotent way.

An alternative used in the present document is to allow the addressing of data inside the REST resource using principles in the spirit of REST, that is, by allowing REST primitives to operate on parts of the data representation. This is an approach popularized by XCAP, which does allow arbitrary xpath expressions to manipulate parts of the XML representation of a resource. The approach taken here is very similar. I.e. the parts of the resource are identified using normal URIs, where the components correspond to the names of the attributes.

In this way attributes can be DELETEd (e.g. HTTP DELETE), UPDATEd (e.g. HTTP PUT), CREATEd (e.g. HTTP PUT), values can be added to collections (e.g. HTTP POST). And of course the value of a single attribute can be obtained (e.g. HTTP GET).

Partial addressing uses the following conventions:

<resourceURI>/<attributeName> -- addresses the attribute with name *<attributeName>* contained in the addressed resourceURI.

E.g. *<sclBase>/scls/<scl>/searchStrings.*

Addresses the list is of searchStrings in the resource *<sclBase>/scls/<scl>*.

<resourceURI>/<attributeName>[/<subElement>]* -- addresses the data element with name *<subElement>* inside the attribute with name *<attributeName>* inside the addressed *<resourceURI>*.

- If the attribute identified by *<attributeName>* is a primitive type (e.g. DateTime) or an enumeration type then this type of access does not apply
  E.g. *<sclBase>/scls/<scl>/creationTime.*

- If the attribute identified by *<attributeName>* is a complex type with multiple elements, the name of the element is the *<subElement>* component.
  E.g. *<sclBase>/scls/<scl>/subscriptions/<subscription>/filterCriteria/lastModifiedSince.*

- If the attribute identified by *<attributeName>* is a complex type with a single element that is a collection of a simple type or enumeration, the *value* of the element is the *<subElement>* component identifier. If the simpleType is AnyURI, the URI encoded value is used.
  E.g. *<sclBase>/scls/<scl>/searchStrings/tag1.*

- If the attribute identified by <*attributeName*> is a complex type with a single element which is a collection of elements of complex type, then <*subElement*> component identifier is the subElement called "id" of the element in the collection.
  E.g. <*sclBase*>/*accessRights*/<*accessRight*>/*permissions*/<*permission*>
  This way of addressing allows multiple levels. For the next level the same rules apply.
  Or 2 levels deep;
  E.g. <*sclBase*>/*accessRights*/<*accessRight*>/*permissions*/<*permission*>/*permissionFlags*/*READ*
  This means that in order to allow multiple levels of addressing elements of complex types that are not leaves (i.e. contain elements of other complex types) shall have a subElement called id.

9.3.2.29.2 Retrieve an attribute or part of an attribute

The procedure is used to retrieve a specific attribute or part of an attribute.

Issuer: requests to retrieve an attribute in a resource. The request addresses the URI of the resource and the partial access address identifying the attribute or part thereof, i.e. <*resourceURI*>/<*attributeName*>*[/<subElementName>]** as described in clause 9.3.1.3.

Hosting SCL: The validation as defined in the appropriate procedure for the retrieve of the specific resource shall apply as well to the retrieval of a specific attribute or part thereof. In particular, if the issuer has the READ permission on the <resourceURI>, it also has the permission to retrieve a specific attribute or part of a specific attribute.

The permission of a resource applies to attributes within the resource, however the attribute-type shall be considered for the proper authorization.

After doing the actions specified in the normal retrieve procedure on the resource, the hosting SCL shall respond to the issuer with the appropriate generic responses as indicated in clause 9.3.1.1, it shall return the value of the specified attribute or part of the specified attribute.

If the attribute does not exist (e.g. because no value was provided for an optional attribute) a "not found" error is returned.

9.3.2.29.3 Delete an attribute

The procedure is used to delete a specific attribute, which may lead to the hosting SCL providing a default value for the attribute.

Issuer: requests to delete an attribute in a resource. The request addresses the URI of the resource and the partial access address identifying the attribute, i.e. <*resourceURI*>/<*attributeName*> as described in clause 9.3.1.3.

Hosting SCL: The validation as defined in the appropriate procedure for the modify procedure of the specific resource shall apply as well to the delete of a specific attribute. In particular, if the issuer has the WRITE permission on the <*resourceURI*>, it also has the permission to delete a specific attribute.

The SCL shall handle to the request according to Table 9.63.

Table 9.63

Mandatory in modify request	Mandatory in resource	Read-Only	Hosting SCL action
NA	Y	Y	Reject the request with a "forbidden" error.
N	Y	N	Revert the attribute to a default value.
Y	Y	N	Reject the request with "forbidden" error.
N	N	N	Remove the attribute from the resource representation.

If the attribute is WO, then the request shall be rejected with a "forbidden" error.

After doing the actions specified in the normal modify procedure on the resource, the hosting SCL shall respond to the issuer with the appropriate generic responses as indicated in clause 9.3.1.1.

If the addressed attribute did not exist (e.g. because no value was provided for an optional attribute) a "not found" error is returned.

9.3.2.29.4 Delete a part of an attribute

This procedure is used to modify an attribute by deleting a part of its structured value or an element from its collection.

Issuer: requests to delete part of an attribute in a resource. The request addresses the URI of the resource and the partial access address identifying the a subElement in an attribute, i.e. *<resourceURI>/<attributeName>[/subElement]** as described in clause 9.3.1.3.

Hosting SCL: The validation as defined in the appropriate procedure for the modify procedure of the specific resource shall apply as well to the delete of a part of an attribute. In particular, if the issuer has the WRITE permission on the *<resourceURI>*, it also has the permission to delete a part of a specific attribute.

If the attribute is WO, then the request shall be rejected with a "forbidden" error.

If the addressed subElement is an optional element in its parent (i.e. with cardinality [0..1]) then the subElement shall be deleted.

If the addressed subElement is an mandatory element in its parent (i.e. with cardinality [1..1]) then the delete shall be rejected with a "forbidden" error response.

If the addresses subElement is an element in a collection parent (i.e. with cardinality [0..unbounded]) then the element shall be deleted from the collection.

After doing the actions specified in the normal modify procedure on the resource, the hosting SCL shall respond to the issuer with the appropriate generic responses as indicated in clause 9.3.1.1.

If the addressed attribute did not exist (e.g. because no value was provided for an optional attribute) a "not found" error is returned.

9.3.2.29.5 Replace a attribute or part of an attribute

Issuer: requests to replace the value of an attribute or part of an attribute in a resource with a new value. The request addresses the URI of the resource and the partial access address identifying the attribute or part thereof, i.e. *<resourceURI>/<attributeName>[/subElement]** as described in clause 9.3.1.3. The request contains the new value of the attribute or subElement.

Hosting SCL: The validation as defined in the appropriate procedure for the modify procedure of the specific resource shall apply as well to the replace of an attribute or part thereof. In particular, if the issuer has the WRITE permission on the *<resourceURI>*, it also has the permission to replace a value of a specific attribute or part thereof.

The permission of a resource applies to all attributes within the resource, however the attribute-type shall be considered for the proper authorization.

If the attribute is an RO or WO attribute, the request shall be rejected with a "forbidden" error.

If the addressed attribute did not exist (e.g. because no value was provided for an optional attribute) the replace shall be interpreted as an add/create of the attribute or subElement, provided the attribute or subElement constitutes a valid attribute or element according to the attributes description.

If the attribute, or part thereof, are not a valid attribute (e.g. *<resourceURI>/someNonExistingAttribute*) then a "Not Found" shall be returned.

After doing the actions specified in the normal modify procedure on the resource, the hosting SCL shall respond to the issuer with the appropriate generic responses as indicated in clause 9.3.1.1.

9.3.2.29.6 Add new values to collection attribute

Add a value or multiple values to a collection.

Issuer: requests to add a new value to an attribute or subElement collection in a resource. The request addresses the URI of the resource and the partial access address identifying the attribute or part thereof, i.e. *<resourceURI>/<attributeName>[/subElement]** as described in clause 9.3.1.3. The addressed attribute of subElement shall be a collection type. This means that it is complex type which only has one subElement with a cardinality of [0..unbounded]. The request contains a single value or a list of values that shall be added to the collection.

Hosting SCL: The validation as defined in the appropriate procedure for the modify procedure of the specific resource shall apply as well to the delete of a part of an attribute. In particular, if the issuer has the WRITE permission on the <*resourceURI*>, it also has the permission to add new values to a collection of a specific attribute or subElement.

If the addressed attribute or subElement is not a collection then the request shall be refused with a "method not supported" error response.

If the addressed attribute or subElement is a collection (i.e. with multiplicity [0..unbounded]) then the provided value or values shall be added to the collection.

After doing the actions specified in the normal modify procedure on the resource, the hosting SCL shall respond to the issuer with the appropriate generic responses as indicated in clause 9.3.1.1.

If the addressed attribute did not exist (e.g. because no value was provided for an optional attribute) a "not found" is returned.

9.3.2.29.7 Subscribe to an attribute or part of an attribute

The client subscribes to an attribute or a sub-attribute and is only notified when that specific attribute changes.

Issuer: requests to subscribe to attribute or subElement in a resource. The request addresses the *subscriptions* child of the resource. The filtercriteria identify the attribute or part thereof, i.e. <*attributeName*>*[/subElement]** as described in clause 9.3.1.3.

Hosting SCL: The validation as defined in the appropriate procedure for the subscribe procedure of the specific resource shall apply. In addition, the hosting SCL shall check that the addresses attribute or subElement exists. If it does not exist, the request is rejected with a "invalid request" error.

After doing the actions specified in the normal subscribe procedure on the resource, the hosting SCL shall respond to the issuer with the appropriate generic responses as indicated in clause 9.3.1.1.

9.3.2.29.8 Notify on an attribute or part of an attribute

The server notifies a client about a change to an attribute or a sub-attribute. As a precondition, the issuer of the subscription should have created a subscription with filter criteria that include a attribute accessor as described above.

Hosting SCL: notifies the subscriber about the change of a resource matching the specified filter criteria as described in clause 9.3.2.19.6. Since the filtercriteria define a path (or attribute accessor), only modifications of the resource that are hierarchically sub-ordinate to the indicated path will be notified.

The notify will contain only that part of the resource structure that would be present in an equivalent partial retrieve using the same attribute accessor.

However, if the attribute or sub-attribute is deleted that is referred to as attributeAccessor in the filtercriteria, then the hosting SCL shall notify the subscriber with a error status of "not found" and terminate the subscription.

Issuer: The actions of the issuer are defined in clause 9.3.2.19.6.

9.3.2.30 Collection management

9.3.2.30.1 Introduction

This clause describes the procedures for managing collections in general. In clause 9.2 there are several collections indicated in the resource structure, for example *scls*, *applications*" to name only few of them, for a detailed list of the collection, see the resource structure in clause 9.2. Collections shall not be explicitly created via one of the reference point (mIa, dIa and mId), they shall be implicitly created when their parent is created, therefore they shall not be deleted either. It shall only be possible to modify the content of the collections (by adding and removing resources) or to read the content of the collection.

9.3.2.30.2 Read all resources in a collection

This procedure shall be used for getting a list of references to all child resources in the addressed collection resource.

Issuer: requests to retrieve the information of a collection resource. The request shall address one of the collection resource defined in clause 9.2.

Hosting SCL: shall validate the received request. Retrieval shall only be allowed if the issuer is authorized to retrieve the collection resource, according to the accessRight defined for the collection resource where the request is targeted. The hosting SCL shall return a successful response that contains the representation of the collection resource, with all the attributes defined in the collection resource and a list representing the URIs of all the child resources for which the issuer has some/any permission. In order to make a resource visible without actually given permission to access resource in any way, the DISCOVERY permission flag can be used.

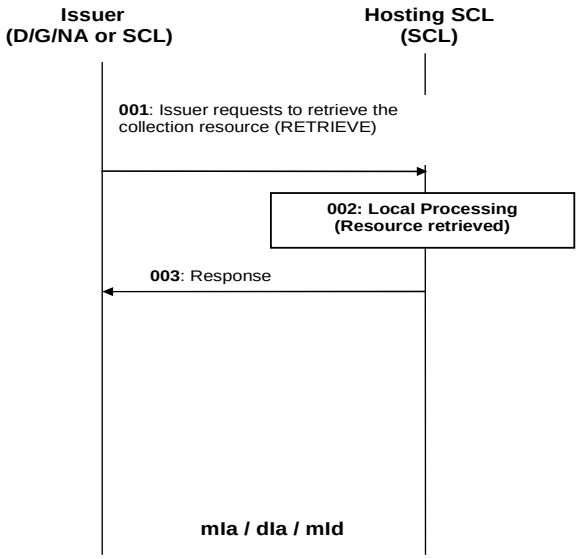


Figure 9.122: Procedures for collection retrieve

Step 001: The Issuer requests to the Hosting SCL to retrieve a collection resource.

Step 002: The Hosting SCL shall check if the Issuer is authorized to retrieve the collection resource. More details about the behaviour of the Hosting SCL are provided above.

Step 003: The Hosting SCL responds positively to the request.

List of main procedure specific exceptions:

Step 002: The Issuer is not authorized to retrieve the resource. The Hosting SCL responds with an error.

9.3.2.30.3 Update collection attributes

This procedure shall be used for modifying the attributes defined in the collection resource.

Issuer: requests to modify all or some of the attributes of an application collection resource, by using an UPDATE. The request shall address either:

- a collection resource as defined in clause 9.2, the issuer shall send new (proposed) values for all mandatory read-write attributes and may send values for the optional read-write attributes; or
- by modifying (setting a value) of a single attribute of the collection resource; by providing the new proposed value; or
- by adding or deleting an attribute of the collection resource.

Hosting SCL: shall validate the received request. Update shall only be allowed if the issuer is authorized to modify the collection resource, according to the accessRight defined for the collection resource itself. The hosting SCL shall then modify the resource representation according to the modifications. The hosting SCL may modify some of the attributes, e.g. expiration time, due to internal policies. In case that the issuer requested to delete one of the optional attribute, the hosting SCL shall provide default values for attributes that are required by the SCL.

The hosting SCL shall return a generic response according to clause 9.3.1.1.

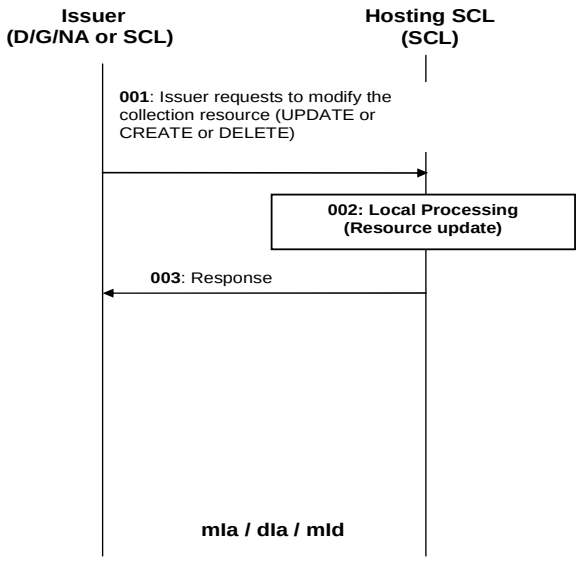


Figure 9.123: Procedures for collection update

Step 001: The Issuer requests the Hosting SCL to modify a collection resource. The Issuer has three options for updating the resource as indicated in the text above. The Issuer can therefore use an UPDATE, CREATE or a DELETE.

Step 002: The Hosting SCL shall check if the Issuer is authorized to perform the modification to the collection resource. The SCL shall perform the changes to the resource. More details about the behaviour of the Hosting SCL are provided above.

Step 003: The Hosting SCL responds positively to the request.

List of main procedure specific exceptions:

Step 002: The Issuer is not authorized to modify (update, create or delete) the resource. The Hosting SCL responds with an error.

9.3.2.30.4 Add a child resource to a collection

This procedure shall be used for modifying the content of a collection resource by adding a new child resource. The procedure is described in the following clauses depending on the type of resource that need to be added (e.g. <*scl*>, <*application*>, <*container*> and so on).

9.3.2.30.5 Delete a child resource from a collection

This procedure shall be used for modifying the content of a collection resource by removing a child resource. The procedure is described in the following clauses depending on the type of resource that need to be deleted (e.g. <*scl*>, <*application*>, <*container*> and so on).

9.3.2.30.6 Subscription management for a collection

These procedures shall be used for subscribing for changes in a collection resource and managing the subscription itself. The procedures are described in detail in clause 9.3.2.18.

9.3.2.31 SCL retargeting mechanism

9.3.2.31.1 Introduction

One of the functions of the SCL is to route ETSI M2M messages between M2M applications and other M2M capabilities, after proper verification of relevant access rights. This behaviour requires that the application register an Application Point of Contact (see "aPoC" attribute in the <*application*> resource tree). If an application does not provide an Application Point of Contact, the SCL retargeting behaviour is not available.

In addition, if the aPoC attribute is specified, then the application shall also publish a set of Application Point of Contact Paths ("aPoCPaths" attribute), which restricts the SCL retargeting behaviour to only resources addressed by specific sub-paths. The Application Point of Contact Paths are relative to the URI of to the path already registered by the application registration resource.

The depth of the tree of resources that can be addressed using the retargeting is controlled by the *aPocHandling* attribute in the <*sclBase*> resource that is negotiated during registration or re-registration for G/DSCL (see create <scl> and update <scl>) or configured for a NSCL. The default handling shall always be 'SHALLOW' which means that each application resource that can be addressed using the SCL retargeting mechanism shall be explicitly specified in the aPoCPaths attribute. The exception to this rule, is for large tables, where rows in the table can be addressed without having to specify each row as a separate resource in the aPoCPaths attribute, therefore, it is allowed to address one level deeper than the aPoCPaths indicate to handle the SHALLOW case.

In case the aPocHandling is configured or negotiated as "DEEP" then any resource 'behind' the specified paths in the aPoCPath attribute can be addressed.

Each Application Point of Contact Path is a tuple containing:

- A path relative to the URI of the application path resource.
- An optional Access Right identifier., which This applies to all subresources subordinate URIs of under this path. In the absence of such Access Right identifier, the Access Rights of the application registration resource apply.
- An optional searchStrings element, which is used during the discovery.

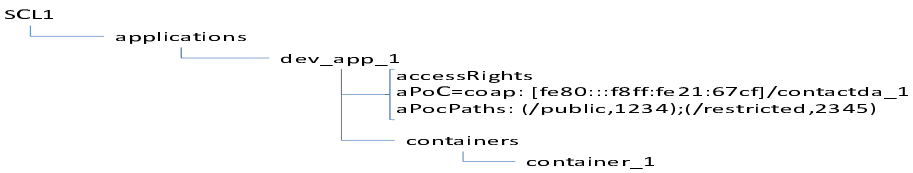


Figure 9.124: Procedures for retargeting

SCL retargeting happens when the requested resource is not stored in the SCL:

- in case of shallow handling:
  - if the absolute target resource URI (i.e. the combination of host header and request URI) either matches the URI of the registered application resource, followed by the path element of one of the *aPoCPaths or* if it is prefixed by said combination with the limitation that there is only one path element following the matched prefix.

- in case of deep handling:
  - if the absolute target resource URI (i.e. the combination of host header and request URI) is prefixed by the URI of the registered application resource, followed by the path element of one of the aPoCPaths.
- when an Access right identifier has been associated to the best matching path element of the *aPoCPaths* (i.e. longest prefix match), retargeting happens if this Access Right resource grants the requestor access to the target resource. If no accessRightID is associated with the best matching path, the accessRightID of the application registration resource is used to determine if the requestor is allowed to access the target resource.

SCL retargeting is based on the target URI of the request and substitutes the resource URI of the registered application path prefix (as registered by the application to its SCL e.g. `/http://gsc1SCL1/applications/dev_app_1/` by the aPoC. The *aPocPaths* is not used for the replacement, but only to determine access rights that apply for accessing the resources.

Some examples based on the above resources:

- if the target URI is `http://SCL1/applications/dev_app_1/some_none_existing_resource`, a "not found" error is returned.
- if the target URI is `http://SCL1/applications/dev_app_1/containers/container_1`, no retargeting is done and resource container_1 on the SCL is returned.
- if the target URI is `http://SCL1/applications/dev_app_1/public/bla`, the request is retargeted to `coap:[fe80:::f8ff:fe21:67cf]/contactda_1/public/bla`, i.e. the `http://SCL1/applications/dev_app_1` is replaced by `coap:[fe80:::f8ff:fe21:67cf]/contactda_1`.
  The retargeting is only allowed if the requestor is allowed to perform the requested method according to the accessRightID associated with best matching aPocPath, which in this case is /public, and therefore accessRightID 1234 is used in the authorization of the request.
- If the negotiated apoc handling is "SHALLOW", then if the target URI is `http://SCL1/applications/dev_app_1/public/bla/anotherlevel`, the request is rejected since only one path element may follow the matched prefix in case of shallow handling
- If the negotiated apoc handling is "DEEP", then if the target URI is `http://SCL1/applications/dev_app_1/public/bla/anotherlevel`, the request is retargeted to `coap:[fe80:::f8ff:fe21:67cf]/contactda_1/public/bla/anotherlevel`, i.e. the `http://SCL1/applications/dev_app_1` is replaced by `coap:[fe80:::f8ff:fe21:67cf]/contactda_1`.
  The accessRightID 1234 applies as before.

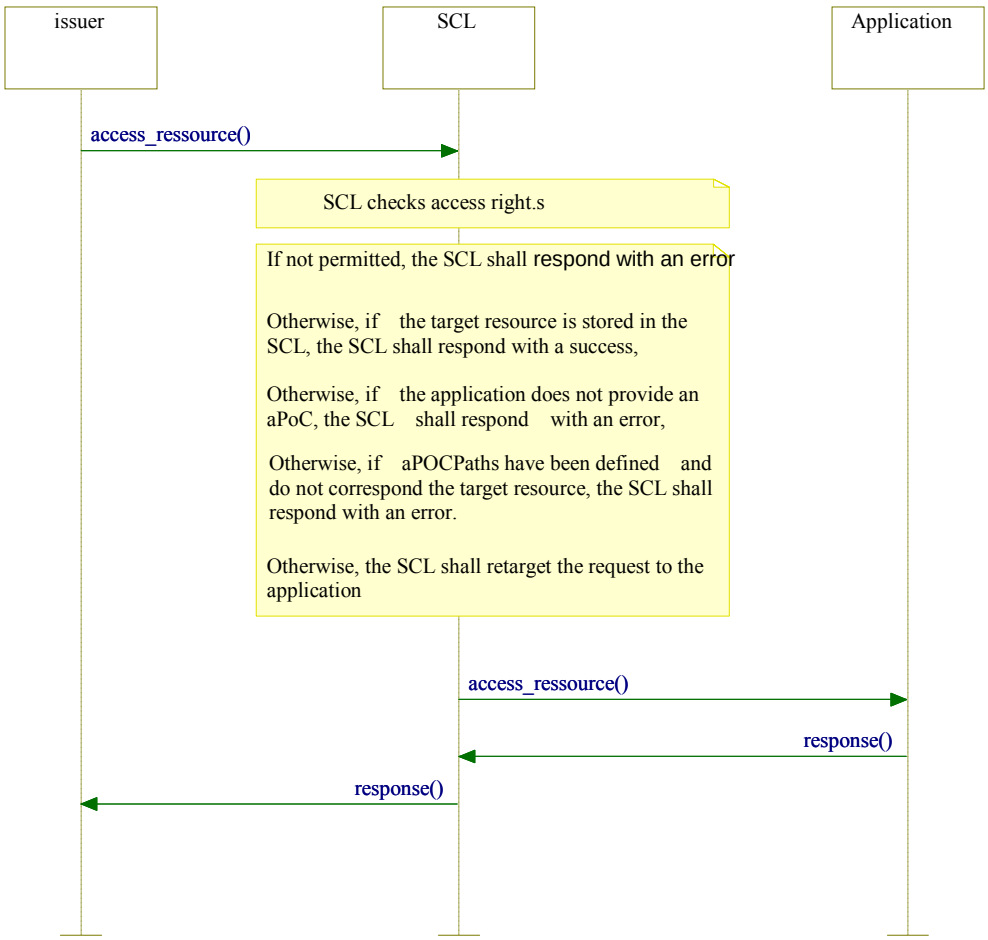


Figure 9.125: Procedures for retargeting

Annex A (informative): Access Network Consideration within the M2M Framework

This Annex provides background information about access networks types and their impacts on the M2M framework within the context of connectivity.

A.1 Void

A.2 Void

A.3 Void

A.4 Void

A.5 Access Networks and the M2M PoC

The M2M PoC holds the information used by the M2M system to route to an SCL. This information is provided to the M2M system by the registered SCL at registration time. The routing to an SCL and ultimately to an application in the M2M system depends on the characteristics of the access network which impacts the information to be conveyed in the M2M PoC. A non-exhaustive number of typical factors to be considered with the various access networks are listed below.

SCLs belonging to Devices handled in a Single Mobile Access Network

SCLs associated with M2M transport devices handled by a single mobile access network have the following properties:

- The Mobile Network takes care of changes in the device point of attachment to the network and offers the address of a single point of attachment.
- M2M transport device IP address can change even when its point of attachment to the network does not change.

SCLs belonging to Devices handled by a Single Fixed Network

SCLs associated with transport devices handled by a single fixed access network have the following properties:

- Its point of attachment to the network is fixed.
- It can have a fully qualified domain name that resolves to the IP address of the fixed SCL, or a fixed IP address allocated to the fixed SCL (e.g. M2M gateway).
- The IP address can be private or public.

SCLs belonging to Devices handled by multiple Access Networks

SCLs associated with dual access M2M transport devices handled by multiple access networks have the following properties:

- The M2M transport device could de-register from the old access network and register with the new access network. Otherwise the M2M transport device could be registered simultaneously in both access networks.
- The M2M transport device will have a different IP address associated with each access network.

As can be seen, the routing to an SCL depends on the characteristic of the access network. In general the easiest routing to an SCL is achieved when a static public IP address is assigned to M2M device, as it possible to rely on direct DNS address translation or dynamic DNS address translation.

Considering the above aspects, the M2M PoC for an M2M registered SCL can have a URI conforming to RFC 3986 [38] as follows:

URI = scheme:/fullyqualifieddomainname/path/; or

URI=scheme://ip-address/path/.

A.5.1 M2M PoC for M2M SCLs associated with a Fixed Network

The IP address allocated to an SCL can be static and public, public, or private. In all cases, the M2M PoC includes a static public IP address or a fully qualified domain name that can be resolved to a public IP address. If the IP address is private, then the public IP address of the PPP is used.

A.5.2 M2M PoC for M2M SCLs associated with Mobile Network

If the IP address for the registered, SCL cannot be reliably used, and as such cannot be included in the M2M PoC, then in this case the M2M PoC for the registered SCL includes appropriate information as defined by various access networks.

Each access network specifies the means for allowing an M2M SP to fetch the IP address associated with an SCL attaching to that access network and consequently the information to be included in the M2M PoC for the registered SCL.

In the event that the M2M SP has connections to multiple access networks, there is a need to establish a binding between the registered SCL and the access network. That binding can be established through SCLs explicitly listing the access network at registration/update time, otherwise the M2M SP can derive it (using the link over which the registration arrived) and storing it and binding it to the registration information.

Two options have to be considered in this case. The first option assumes that the M2M transport device associated with the M2M SCL has an IP address allocated to it, while the second option assumes that M2M transport device associated with the M2M SCL does not have an IP address allocated to it For both options, the access network over which the registration from the M2M transport device arrives are recorded (optionally it can be included within the M2M PoC) for the impacted M2M SCL by the M2M system and used later for delivery to the target M2M transport device associated with the M2M SCL.

Option 1: M2M Transport Device has an IP address

In this option, the M2M SP will perform the actual delivery. To that effect, the M2M SP will have to acquire the most recent IP address allocated to the target M2M transport device associated with the M2M SCL from the access network (relying on mechanisms defined by the various access networks).

Option 2: M2M Transport device does not have an IP address

If the M2M SP discovered that the M2M transport device does not have an IP address allocated to it (through same techniques in option1), then it has two sub-options. First sub-option assumes that SMS is not available, and as such the M2M SP needs to delegate to the access network the task of paging the M2M transport device so that it can acquire an IP address to establish a transport connection for communication with the M2M SP.

The manner in which such a delegation is achieved is access network dependent and will be defined by the various access networks.

In the second sub-option, SMS can be used by the M2M SP to wake up the M2M application so it can acquire an IP address to establish a transport connection to the M2M SP. Subsequently the M2M SP can fetch the IP address and proceed as is the case in option 1.

A.5.3 Routing to M2M SCLs associated with multiple access networks

When the M2M transport device attaches to a network, the M2M PoC for a registered M2M SCL conforms to the procedures associated with a fixed network.

When the M2M transport device attaches to a mobile network, the M2M PoC for a registered M2M conforms to the procedures associated with mobile networks.

If an M2M transport device already attached to an access network attaches to a new access network, the M2M SCL updates its reachability data.

Annex B (normative): <mgmtObj> Resource Instances Description

B.1 M2M Management Function List

This clause provides all functions that are required in an ETSI M2M system for Remote Entity Management. Such functions are provided by ETSI M2M specific management resources and/or the management resources mapped from the equivalent data models of OMA-DM and BBF TR-069 as shown in Table B.1.

The M2M management functions are divided into following packages:

- **General Management (GEN):** Allows retrieving general information of the M2M Device or Gateway, and provides generic mechanism applicable to different specific management functions.
- **Configuration Management (CFG):** Allows configuration of the device capabilities and features for supporting M2M services and applications, including activating / deactivating device hardware components or I/Os in the M2M Device or Gateway.
- **Diagnostic & Monitoring Management (D&M):** Allows running specific diagnostic tests on a device and collecting the results or alerts from the M2M Device or Gateway. This package is also called Fault and Performance Management.
- **Software/Firmware Management (SFW):** Allows installation /update/removal of application specific or SCL related software / firmware in M2M Device or Gateway.
- **Area Network Management (ANW):** Allows M2M Gateway-specific configuration and M2M Area Network and Device management through a M2M Gateway.
- **SCL Management (SCL):** Allows remote configuration and retrieval of M2M Device or Gateway service capability layer parameters.

Although the ETSI M2M specific data model SHOULD be based on the "common" part of existing OMA-DM and TR-069 data models, it MAY contain a management object/parameter which is missing from one existing model or another. The bottom line is that the management object/parameter SHALL be supported by at least one existing data model. Any request for the missing object/parameter of a M2M Device/Gateway which implements only OMA-DM or TR-069 model is still allowed, but SHALL result in an error response (e.g. 404 not found).

ETSI M2M specific data model shall be supported as mandatory function, while others are optional at implementation discretion.

All management functions except for SCL Management are exposed to network applications via mIa reference point. The access to those functions is controlled by the accessRight resource associated with the corresponding management resources.

The same Management Object parameter on the same device shall not be allowed to be modified by more than one management authority (e.g. NA).

Table B.1: Remote Entity Management Function List

ID	Function	ETSI resources and procedures	Description	Equivalent in OMA DM	Equivalent in BBF TR-069
GEN-001	Retrieve device manufacturing information	etsiDeviceInfo	The device manufacturing information includes device identifier, device type, manufacture information, version information, etc.	urn:oma:mo:oma-dm-devinfo:1.0 [13] urn:oma:mo:oma-dm-devdetail:1.1 [13]	DeviceInfo [14]
GEN-002	Enable/Disable a specific REM function	CREATE/ DELETE corresponding <*mgmtObj*>	Enable or Disable a specific REM function to be exposed via mIa by creating or deleting the corresponding <*mgmtObj*> resource in NSCL. The use cases include but not limited to adding a instance for a multi-instance manageable parameters/ objects (e.g. firmware/ software), and activate a management function only when it is needed (e.g. disable wifi/camera when entering a restrict area)	Add / Delete method [8]	AddObject / DeleteObject method [10]
GEN-003	Discover manageable objects/ parameters	Discover the <*mgmtObj*>	Discover the manageable objects / parameters supported by the device at the run-time, especially for the "optional" or "multi-instance" ones	Get MO tree with attribute filter criteria [15]	GetParameterNames method [10]
GEN-004	Support RPC command execution	<*mgmtCmd*>	Performing RPC-like management command without an associated MO tree	n/a	FactoryReset method [10] Reboot method [10] Upload method [10] Download method [10] ScheduleDownload method [10] ScheduleInform method [10] ChangeDUState method [10]
CFG-001	Activate / deactivate device hardware components. And retrieve current status	etsiDeviceCapability	The hardware components include camera, display, speaker, USB, GPS, etc.	urn:oma:mo:oma-dcmo:1.0 [16]	UserInterface [17] SmartCardReader [17] USBHosts [17]

ID	Function	ETSI resources and procedures	Description	Equivalent in OMA DM	Equivalent in BBF TR-069
CFG-002	Activate / deactivate communication modules. And retrieve current status	etsiDeviceCapability	The communication modules include Bluetooth®, WLAN, UPnP, etc.	urn:oma:mo:oma-dcmo:1.0 [16]	UPNP [17] WLAN-Configuration [19]
CFG-003	Reboot complete device	etsiReboot	Support reboot level and timing control	urn:oma:mo:oma-diag:restart:1.0 [18]	Reboot method [10]
CFG-004	Factory reset complete device	etsiReboot	Reset to factory default state	urn:oma:mo:oma-lawmo:1.0 [20]	Factory reset method [10]
D&M-001	Monitor general device run-time parameters	n/a	General run-time parameters include device uptime, roaming statue, etc.	urn:oma:mo:oma-dm-devdetail:1.1 [13]	DeviceInfo [14], [17]
D&M-002	Monitor battery information	etsiBattery	Battery information includes current level, charging status, standby time, etc.	urn:oma:mo:oma-diag:batteryinfo:1.0 [18]	n/a
D&M-003	Monitor memory status	etsiMemory	Memory status includes the available space, total space, etc.	urn:oma:mo:oma-diag:memory:1.0 [18]	DeviceInfo [17]
D&M-004	Monitor CPU and process information	n/a	CPU and process information includes the current usage of CPU, process id, etc.	n/a	DeviceInfo [17]
D&M-005	Monitor temperature information	n/a	The temperature information of the device	n/a	.DeviceInfo [17]
D&M-006	Monitor radio metrics	n/a	The working performance of the radio interface(s) on the device	urn:oma:mo:oma-diag:RFParms_3GPP_GSM:1.0 [18] urn:oma:mo:oma-diag:RFParms_3GPP_UMTS:1.0 [18] urn:oma:mo:oma-diag:RFParms_3GPP_LTE:1.0 [18]	n/a
D&M-007	Monitor NFC status	n/a	NFC parameters	urn:oma:mo:oma-diag:NFC:1.0 [18]	n/a
D&M-008	Set diagnostic trap events notification	etsiTrapEvent	Such events include temperature warning, short of memory, application exception, etc.	urn:oma:mo:oma-diagmontrap:1.0 [21] urn:oma:mo:oma-diag:trap-geo:1.0 [22] urn:oma:mo:oma-diag:trap-rxpwr:1.0 [22]	SetParameterAttritutes method [10] (with NotificationChange argument)

ID	Function	ETSI resources and procedures	Description	Equivalent in OMA DM	Equivalent in BBF TR-069
D&M-009	Collect historical or statistical performance/ fault data	etsiPerformanceLog	Such data include device/application crash logs, application usage statistics, network performance, etc.	urn:oma:mo:oma-diagmon:1.0 [23] urn:oma:mo:oma-diag:trapeventlogging: 1.0 [18] urn:oma:mo:oma-diag:paniclog:1.0 [18] urn:oma:mo:oma-diag:appmonlog: 1.0 [18]	DownloadDiagnostics [14], [19] UploadDiagnostics [14], [19] IPPingDiagnostics [14], [19] TraceRouteDiagnostics [14], [19] SelfTestDiagnostics [17] NSLookupDiagnostics [17] PeriodicStatistics [17]
SFW-001	Firmware Management	etsiFirmware	Checking firmware status, performing firmware download, update and removal operations	urn:oma:mo:oma-fumo:1.0 [24]	Download Method [10] Command in Signed Package [10]
SFW-002	Software Management	etsiSoftware	Checking software status, performing software download, (un)install, (de-)activate and removal operations	urn:oma:mo:oma-scomo:1.0 [25]	SoftwareModules [17] ChangeDUState Method [10]
ANW-001	Management of attached D' (d) devices	attachedDevices	List of attached D' (or d) devices and the MOs thereof	urn:oma:mo:oma-gwmo-deviceinventory:1.0 [26]	InternetGatewayDevice LANDevice [19]
ANW-002	Report device attach/detach events	SUBSCRIBE/ NOTIFY procedures	Gateway can report the events of the (de)attachment of D' devices	urn:oma:mo:oma-gwmo-config:1.0 [26]	Active Notification [10]
ANW-003	Fan-out management	Reuse *<group>* resource	A single operation performed on a number of devices. (e.g. bulk firmware upgrade, group-based operations)	urn:oma:mo:oma-gwmo-imageinventory:1.0 [26] urn:oma:mo:oma-gwmo-fanout:1.0 [26]	n/a
ANW-004	Area Network Management	etsiAreaNwkInfo etsiAreaNwkDeviceInfo	Configuring or monitoring the area network specific parameters	n/a	n/a
SCL-001	Registration Configuration/ Retrieval	etsiSclMo	Default parameters/policies related to SCL registration (e.g. target SCL list, expiration duration, accessRightID, searchStrings)	n/a	n/a
SCL-002	Announcement Configuration/ Retrieval	etsiSclMo	Default parameters/policies related to announcement (e.g. announcedTo list, expiration duration)	n/a	n/a

ID	Function	ETSI resources and procedures	Description	Equivalent in OMA DM	Equivalent in BBF TR-069
SCL-003	Misc	etsiSclMo	Other default parameters/policies (e.g. Max number of discovery records or policies for Store-And-Forward handling)	n/a	n/a

B.2 ETSI M2M specific <mgmtObj> resource instances

B.2.1 Resource etsiSclMo

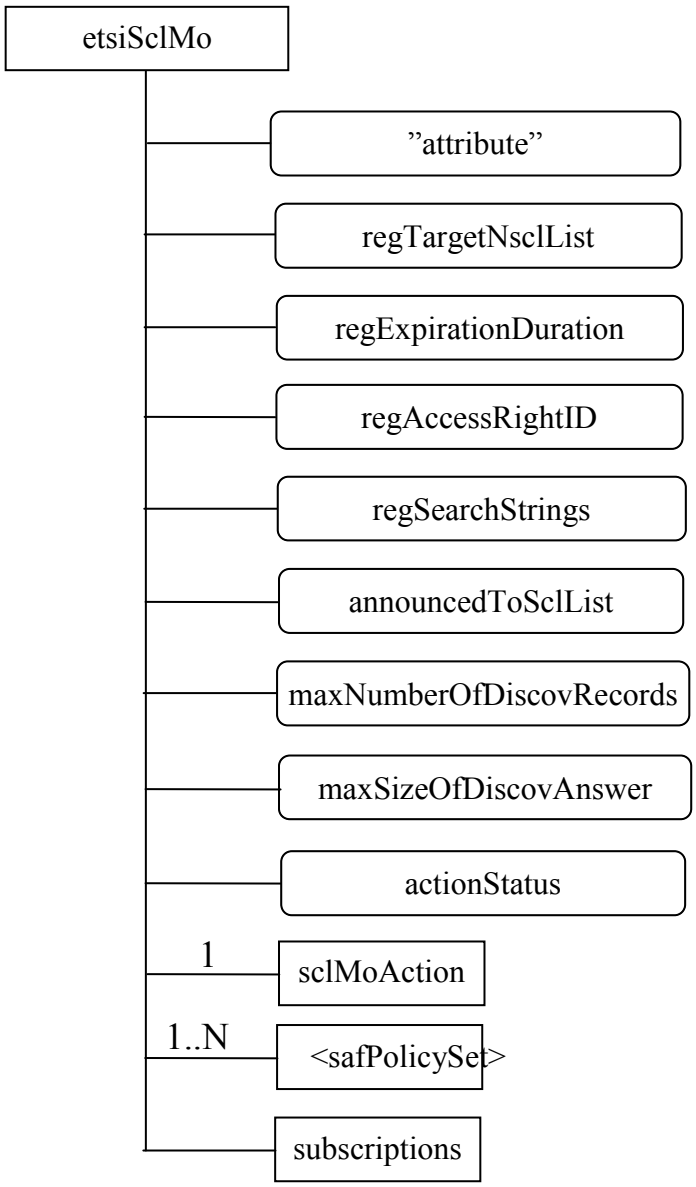


Figure B.1: SCLMO structure

The etsiSclMo resource shall contain the following sub resources:

Table B.2

SubResource	Mandatory/Optional	Multiplicity	Description
subscriptions	M	1	See clause 9.2.3.22.
sclMoAction	M	1	A sub-resource that contains the action to be executed.
<safPolicySet>	M	1..N	One or more sub-resource that contains parameters to describe the policies that govern Store-And-Forward handling according to clause 9.3.1.5. At least one sub-resource of this type is present which contains the default policies. More sub-resources of this type are possible to define policies that are specific to a give request issuer (DA or GA). This sub-resource is defined in clause B.2.1.2.

and the following attributes:

Table B.3

AttributeName	Mandatory/Optional	Type	Description
expirationTime	M	RW	See clause 9.2.2.
accessRightID	M	RW	See clause 9.2.2.
searchStrings	M	RW	See clause 9.2.2.
creationTime	M	RO	See clause 9.2.2.
lastModifiedTime	M	RO	See clause 9.2.2.
moID	M	WO	See clause 9.2.3.27.
originalMO	M	WO	See clause 9.2.3.27.
Description	O	RW	The text-format description of mgmtObj.
regTargetNsclList	M	RW	A list of NSCLs that the D/GSCL may use in order to perform a new registration (after reboot or not).
regExpirationDuration	O	RW	This represents the duration of the next registration. This parameter shall not be set to zero. Any negative value stands for infinite duration.
regAccessRightID	O	RW	The default accessRightID to be allocated to the <scl> when the <scl> is created in the hosting NSCL during the registration procedure of the corresponding D/GSCL.
regSearchStrings	O	RW	This allows to modify the criteria for discovery of D/GSCL.
announcedToSclList	M	RW	Set default announcedTo-SCL list for D/GSCL to announce local resources to remote SCLs (especially when the application does not specify the announceTo attribute when creating resources).
maxNumberOfDiscovRecords	O	RW	Limits the maximum number of resourceURIs contained in a discovery result. This attribute and the maxSizeOfDiscovAnswer attribute are mutually exclusive.
maxSizeOfDiscovAnswer	O	RW	The maximum size of the Discovery Answer. This value is expressed in bytes and is useful for very constrained devices. This attribute and the maxNumberOfDiscovRecords attribute are mutually exclusive.
actionStatus	M	RO	Indicates the status of the Action (including a progress indicator, a final state and a reminder of the requested action).

The etsiSclMo is used by a NSCL in order to configure some parameters in a D/GSCL. The new parameters are taken into account for example after a reboot.

B.2.1.1 sclMoAction resource

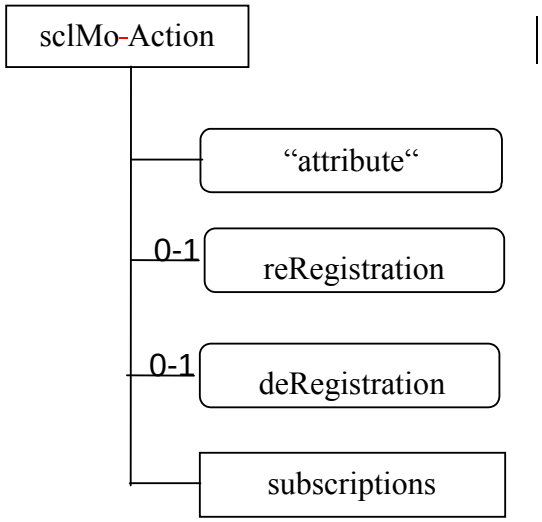


Figure B.2: Structure of the sclMoAction resource

The sclMoAction resource shall contain the following sub resource:

Table B.4

SubResource	Mandatory/Optional	Multiplicity	Description
subscriptions	M	1	See clause 9.3.2.22.

and the following attributes:

Table B.5

AttributeName	Mandatory/Optional	Type	Description
accessRightID	M	RW	See clause 9.2.2.
creationTime	M	RO	See clause 9.2.2.
lastModifiedTime	M	RO	See clause 9.2.2.
reRegistration	O	RW	This action allows to immediately apply the new attributes by triggering a re-registration. NOTE: An explicit de-registration shall be performed by removing the corresponding registered <*scl*> resource, not by sclMo handling.
deRegistration	O	RW	This action triggers the SCL to perform the deregistration procedure.

The execution of the action is triggered by the UPDATE of the corresponding action attribute.

A subsequent action shall not be authorized before the completion of the previous one.

B.2.1.2 <safPolicySet> resource

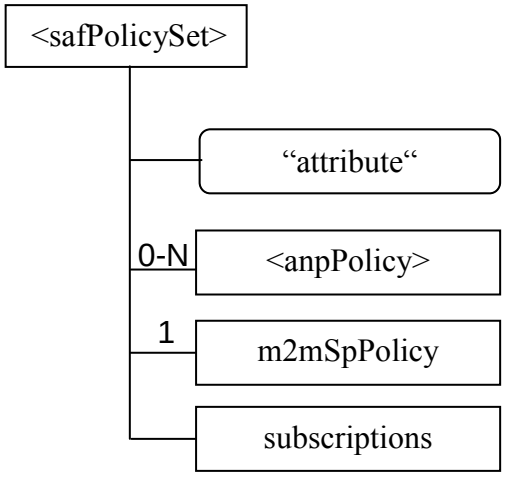


Figure B.3: Structure of the safPolicies resource

The safPolicies resource shall contain the following sub resources:

Table B.6

SubResource	Mandatory/Optional	Multiplicity	Description
<anpPolicy>	M	0..N	Zero or more sub-resource that contain SAF-handling policies which are controlled by the access network provider(s). Depending on what mechanism an access network provider uses to configure policies - either via this MO or with an out-of-scope mechanism - there may be a <anpPolicy> present as a sub-resource. If more than one access network can be used by D/GSCL to establish connectivity, more than one of these sub-resources may be used. For each access network, as identified by the anName in the anpPolicy, there shall be at most one anpPolicy in the set. See clause B.2.1.3 for the definition of <anpPolicy>.
m2mSpPolicy	M	1	A sub-resource that contains SAF-handling policies which are controlled by the M2M service provider. See clause B.2.1.4 for its definition.
subscriptions	M	1	See clause 9.2.3.22.

and the following attributes:

Table B.7

AttributeName	Mandatory/Optional	Type	Description
accessRightID	M	RW	See clause 9.2.2.
creationTime	M	RO	See clause 9.2.2.
lastModifiedTime	M	RO	See clause 9.2.2.
policyScope	M	RW	This attribute defines the scope of the policies contained in this instance of "<safPolicySet>". The policyScope attribute can either be a string with the value "default" which indicates that the policies are the default policies for this SCL or it can be set to a list of APP-IDs in which case the policies are applicable to requests coming from any of the listed applications. There shall be exactly one "<safPolicySet>" resource present with the policyScope attribute set to "default". There shall be at most one "<safPolicySet>" resource present with the policyScope containing a specific APP-ID value.

B.2.1.3 <anpPolicy> resource

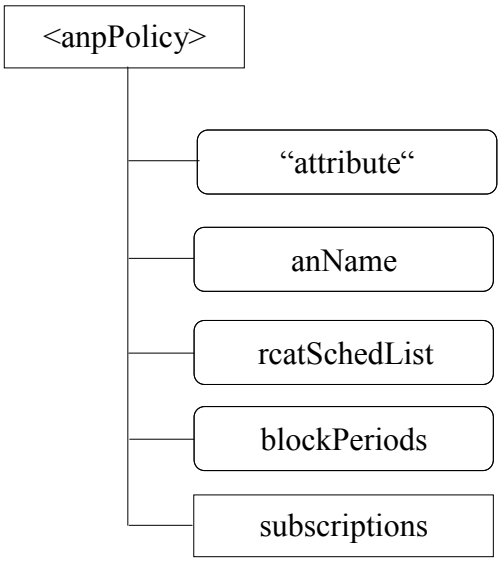


Figure B.4: Structure of the <anpPolicy> resource

The <anpPolicy> resource shall contain the following sub resources:

Table B.8

SubResource	Mandatory/Optional	Multiplicity	Description
subscriptions	M	1	See clause 9.2.3.22.

and the following attributes:

Table B.9

AttributeName	Mandatory/Optional	Type	Description
accessRightID	M	RW	See clause 9.2.2.
creationTime	M	RO	See clause 9.2.2.
lastModifiedTime	M	RO	See clause 9.2.2.
anName	M	RW	An access network name that is used by the managed D/GSCL to determine for which access network the policies defined in this <anpPolicy> resource shall apply. When trying to setup connectivity, the D/GSCL needs to be able to use that name internally to establish connectivity via the associated access network.
rcatSchedList	M	RW	A list of RCAT schedule policy items for defining when it is appropriate to use the access network associated with "anName" for processing requests of specific RCAT values. Each item contains one RCAT value and defines a set of allowed time-spans, where the time spans can be defined as combinations of absolute time spans and reoccurring Schedule time spans. Details of the format for RCAT schedule policy items are specified in [1].
blockPeriods	O	RW	A list of block period policy items for defining how long an D/GSCL shall wait before re-trying to establish connectivity via the access network associated with "anName" after the previous attempt has failed. Each item consists of an number of consecutive failed attempts and a duration. The number of consecutive failed attempts defines how many consecutive attempts for establishing connectivity shall have failed to apply this block policy item and the duration defines how long the next attempt shall be blocked. The D/GSCL shall always apply the block policy item with the largest number of consecutive failed attempts that is smaller or equal to the actual number of consecutive failed attempts. Details of the format for block period policy items are specified in [1].

B.2.1.4 m2mSpPolicy resource

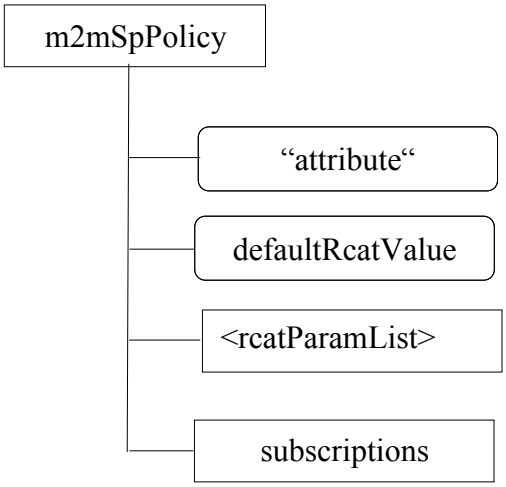


Figure B.5: Structure of the m2mSpPolicy resource

The m2mSpPolicy resource shall contain the following sub resources:

Table B.10

SubResource	Mandatory/Optional	Multiplicity	Description
subscriptions	M	1	See clause 9.2.3.22.
<rcatParamList>	M	N	N sub-resources that contain RCAT-specific parameters for SAF-handling policies. For each possible RCAT value, one of these resources shall exist. In the current release N=8 holds.

and the following attributes:

Table B.11

AttributeName	Mandatory/Optional	Type	Description
accessRightID	M	RW	See clause 9.2.2.
creationTime	M	RO	See clause 9.2.2.
lastModifiedTime	M	RO	See clause 9.2.2.
defaultRcatValue	M	RW	This is the default RCAT value used for requests to remotely hosted resources during SAF-handling when no RCAT value was specified in the request. See clause 9.3.15 for more details of the behaviour description.

B.2.1.5 <rcatParamList> resource

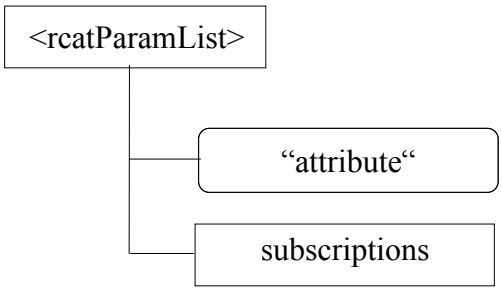


Figure B.5a: Structure of the <rcatParameters> resource

The m2mSpPolicy resource shall not contain any sub resources and the following attributes.

Table B.11b

AttributeName	Mandatory/Optional	Type	Description
accessRightID	M	RW	See clause 9.2.2.
creationTime	M	RO	See clause 9.2.2.
lastModifiedTime	M	RO	See clause 9.2.2.
rcatValue	M	RW	This is the RCAT value with which all parameters in the remaining attributes are associated. Different < rcatParameters > sub resources shall not contain overlapping RCAT values.
defaultTrpdtValue	M	RW	The default TRPDT value to be used for requests of a given RCAT value if no specific TRPDT value is given for the request. Default TRPDT values need to be defined as time durations.
maxPendReqs	O	RW	Threshold for triggering establishment of connectivity by D/GSCL defining the maximum number of pending requests in SAF-handling associated with the given RCAT value. According to the description in clause 9.3.1.5, the D/GSCL shall be allowed to trigger establishment of connectivity when more requests have been stored than the configured maximum number of pending requests for the given RCAT value.
maxPendData	O	RW	Threshold for triggering establishment of connectivity defining the maximum amount of data used for pending requests in SAF-handling associated with the given RCAT value. According to the description in clause 9.3.1.5, the D/GSCL shall be allowed to trigger establishment of connectivity when more data has been stored than the configured maximum amount of data for the given RCAT value**.**
anSelList	O	RW	Ranked list of access network names to govern selection of a preferred access network for processing requests pending in SAF-handling associated with the given RCAT value. According to the description in clause 9.3.1.5, the D/GSCL shall use a ranked list of preferred access networks for a given RCAT value. Each item on that list contains a name in line with the "anName" attributes of <anpPolicy> resources (see clause B.2.1.3) in descending order of preference.

B.2.2 Resource etsiDeviceInfo

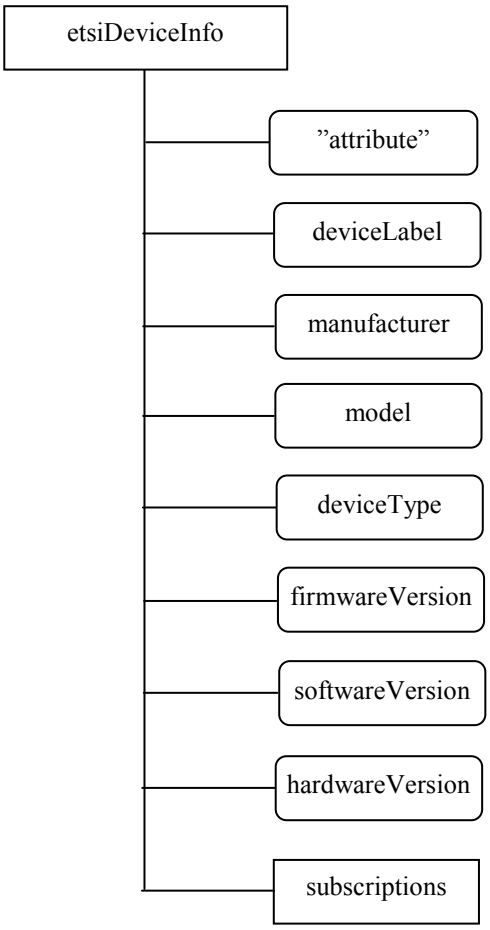


Figure B.6: Structure of the etsiDeviceInfo resource

The etsiDeviceInfo resource shall contain the following sub resources:

Table B.12

SubResource	Mandatory/Optional	Multiplicity	Description
subscriptions	M	1	See clause 9.2.3.22.

and the following attributes:

Table B.13

AttributeName	Mandatory/Optional	Type	Description
expirationTime	M	RW	See clause 9.2.2.
accessRightID	M	RW	See clause 9.2.2.
searchStrings	M	RW	See clause 9.2.2.
creationTime	M	RO	See clause 9.2.2.
lastModifiedTime	M	RO	See clause 9.2.2.
moID	M	WO	See clause 9.2.3.27.
originalMO	M	WO	See clause 9.2.3.27.
Description	O	RW	The text-format description of mgmtObj.
deviceLabel	M	RW	Unique device label assigned by the manufacturer. The uniqueness may be global or only valid within a certain domain (e.g. vendor-wise or for a certain deviceType).
Manufacturer	M	RW	The name/identifier of the device manufacturer.
Model	M	RW	The name/identifier of the device mode assigned by the manufacturer.
deviceType	M	RW	The type (e.g. cell phone, photo frame, smart meter) or product class (e.g. X-series) of the device.
firmware Version	M	RW	The firmware version of the device.
softwareVersion	M	RW	The software version of the device.
hardwareVersion	M	RW	The hardware version of the device.
NOTE: If the device only supports one kind of Software this is identical to softwareVersion.			

B.2.3 Resource etsiDeviceCapability

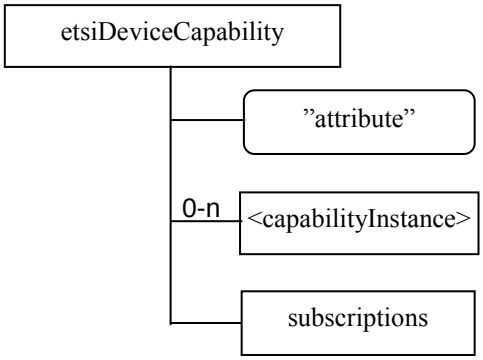


Figure B.7: Structure of the etsiDeviceCapability resource

The etsiDeviceCapability resource shall contain the following sub resources:

Table B.14

SubResource	Mandatory/Optional	Multiplicity	Description
<capabilityInstance>	O	0::n	The sub-resource that contains the information about one of the capabilities of a device (e.g. camera, display, external storage, communication module, etc.).
subscriptions	M	1	See clause 9.2.3.22.

and the following attributes:

Table B.15

AttributeName	Mandatory/Optional	Type	Description
expirationTime	M	RW	See clause 9.2.2.
accessRightID	M	RW	See clause 9.2.2.
searchStrings	M	RW	See clause 9.2.2.
creationTime	M	RO	See clause 9.2.2.
lastModifiedTime	M	RO	See clause 9.2.2.
moID	M	WO	See clause 9.2.3.27.
originalMO	M	WO	See clause 9.2.3.27.
description	O	RW	The text-format description of mgmtObj.

B.2.3.1 <capabilityInstance> resource

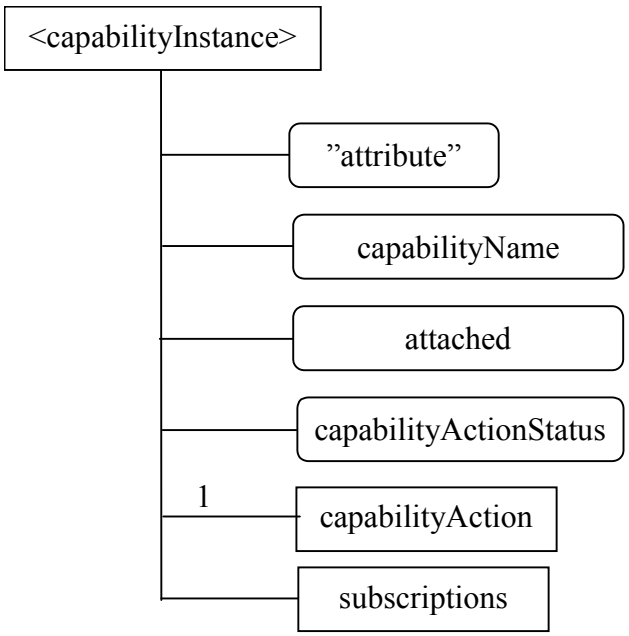


Figure B.8: Structure of the <capabilityInstance> resource

The <capabilityInstance> resource shall contain the following sub resources:

Table B.16

SubResource	Mandatory/Optional	Multiplicity	Description
capabilityAction	M	1	A sub-resource that contains the actions to be executed.
subscriptions	M	1	See clause 9.2.3.22.

and the following attributes:

Table B.17

AttributeName	Mandatory/Optional	Type	Description
accessRightID	M	RW	See clause 9.2.2.
creationTime	M	RO	See clause 9.2.2.
lastModifiedTime	M	RO	See clause 9.2.2.
originalMO	M	WO	See clause 9.2.3.27.
capabilityName	M	WO	The name of the capability.
attached	M	RO	Indicates whether the capability is attached to the device or not.
capabilityActionStatus	M	RO	Indicates the status of the Action (including a progress indicator, a final state and a reminder of the requested action).

B.2.3.2 capabilityAction resource

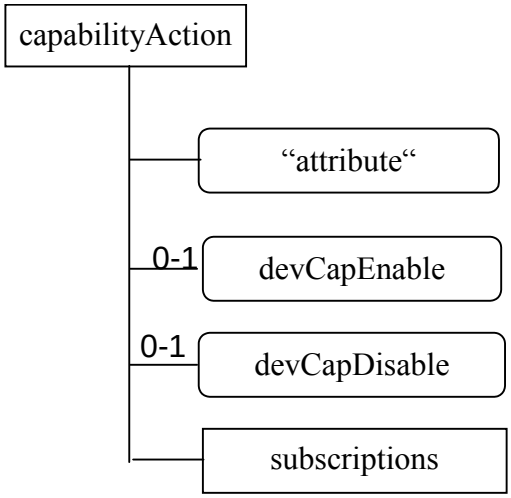


Figure B.9: Structure of the capabilityAction resource

The capabilityAction resource shall contain the following sub resource:

Table B.18

SubResource	Mandatory/Optional	Multiplicity	Description
subscriptions	M	1	See clause 9.2.3.22.

and the following attributes:

Table B.19

AttributeName	Mandatory/Optional	Type	Description
accessRightID	M	RW	See clause 9.2.2.
creationTime	M	RO	See clause 9.2.2.
lastModifiedTime	M	RO	See clause 9.2.2.
devCapEnable	O	RW	The action that allows to enable the device capability.
devCapDisable	O	RW	The action that allows to disable the device capability.

The execution of an action is triggered by the UPDATE of the corresponding action attribute.

A subsequent action shall not be authorized before the completion of the previous one.

B.2.4 Resource etsiBattery

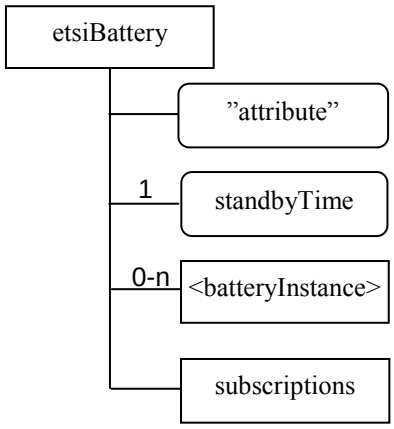


Figure B.10: Structure of the etsiBattery resource

The etsiBattery resource shall contain the following sub resources:

Table B.20

SubResource	Mandatory/Optional	Multiplicity	Description
<batteryInstance>	O	0..n	A sub-resource that contains the information about one of the batteries of a device.
subscriptions	M	1	See clause 9.2.3.22.

and the following attributes:

Table B.21

AttributeName	Mandatory/Optional	Type	Description
expirationTime	M	RW	See clause 9.2.2.
accessRightID	M	RW	See clause 9.2.2.
searchStrings	M	RW	See clause 9.2.2.
creationTime	M	RO	See clause 9.2.2.
lastModifiedTime	M	RO	See clause 9.2.2.
moID	M	WO	See clause 9.2.3.27.
originalMO	M	WO	See clause 9.2.3.27.
description	O	RW	The text-format description of mgmtObj.
standbyTime	M	RW	The estimated time or operation based on the current level of all batteries.

B.2.4.1 Resource <batteryInstance>

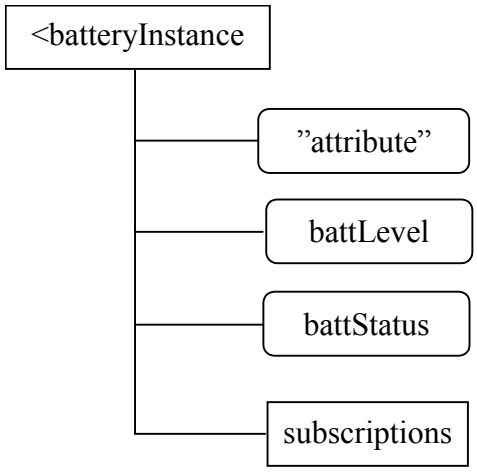


Figure B.11: Structure of the <batteryInstance> resource

The <batteryInstance> resource shall contain the following sub resources:

Table B.22

SubResource	Mandatory/Optional	Multiplicity	Description
subscriptions	M	1	See clause 9.2.3.22.

and the following attributes:

Table B.23

AttributeName	Mandatory/Optional	Type	Description
accessRightID	M	RW	See clause 9.2.2.
creationTime	M	RO	See clause 9.2.2.
lastModifiedTime	M	RO	See clause 9.2.2.
originalMO	M	WO	See clause 9.2.3.27.
battLevel	M	RW	The current battery level.
battStatus	M	RW	Indicates the status of the battery.

B.2.5 Resource etsiMemory

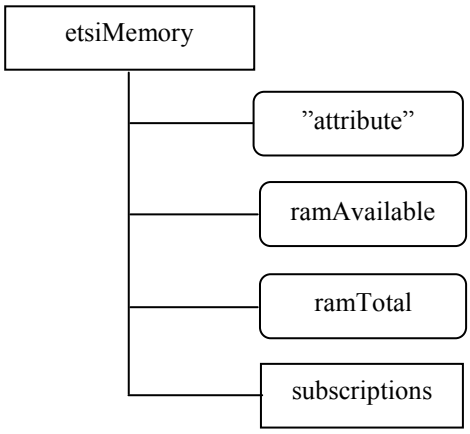


Figure B.12: Structure of the etsiMemory resource

The etsiMemory resource shall contain the following sub resources:

Table B.24

SubResource	Mandatory/Optional	Multiplicity	Description
subscriptions	M	1	See clause 9.2.3.22.

and the following attributes:

Table B.25

AttributeName	Mandatory/Optional	Type	Description
expirationTime	M	RW	See clause 9.2.2.
accessRightID	M	RW	See clause 9.2.2.
searchStrings	M	RW	See clause 9.2.2.
creationTime	M	RO	See clause 9.2.2.
lastModifiedTime	M	RO	See clause 9.2.2.
moID	M	WO	See clause 9.2.3.27.
originalMO	M	WO	See clause 9.2.3.27.
ramAvailable	M	RW	The current available amount of RAM.
ramTotal	M	RW	The total amount of RAM.

B.2.6 Resource etsiTrapEvent

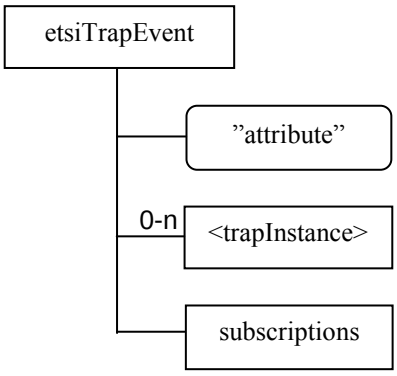


Figure B.13: Structure of the etsiTrapEvent resource

The etsiTrapEvent resource shall contain the following sub resources:

Table B.26

SubResource	Mandatory/Optional	Multiplicity	Description
<trapInstance>	O	0..n	The sub-resource that contains information about one trapEvent supported by a device.
subscriptions	M	1	See clause 9.2.3.22.

and the following attributes:

Table B.27

AttributeName	Mandatory/Optional	Type	Description
expirationTime	M	RW	See clause 9.2.2.
accessRightID	M	RW	See clause 9.2.2.
searchStrings	M	RW	See clause 9.2.2.
creationTime	M	RO	See clause 9.2.2.
lastModifiedTime	M	RO	See clause 9.2.2.
moID	M	WO	See clause 9.2.3.27.
originalMO	M	WO	See clause 9.2.3.27.
description	O	RW	The text-format description of mgmtObj.

B.2.6.1 Resource <trapInstance>

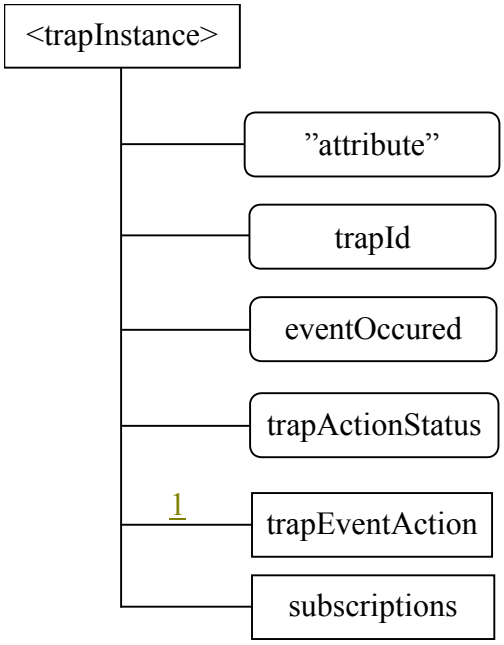


Figure B.14: Structure of the <trapInstance> resource

The <trapInstance> resource shall contain the following sub resources:

Table B.28

SubResource	Mandatory/Optional	Multiplicity	Description
trapEventAction	M	1	A sub-resource that contains the action to be executed.
subscriptions	M	1	See clause 9.2.3.22.

and the following attributes:

Table B.29

AttributeName	Mandatory/Optional	Type	Description
accessRightID	M	RW	See clause 9.2.2.
CreationTime	M	RO	See clause 9.2.2.
LastModifiedTime	M	RO	See clause 9.2.2.
OriginalMO	M	WO	See clause 9.2.3.27.
trapId	M	RW	The identifier that specifies the meaning of the trap event. The content of this attribute is dependent on the underlying management system.
eventOccured	M	RO	Indicated the occurrence of the event. This attribute indicates that the event was raised in the device. The NREM has been informed by the underlying management system that the event was raised in the device and informs the Network Application by using this attribute. The content of this attribute is dependent on the underlying management system.
trapActionStatus	M	RO	Indicates the status of the Action (including a progress indicator, a final state and a reminder of the requested action).

B.2.6.2 trapEventAction resource

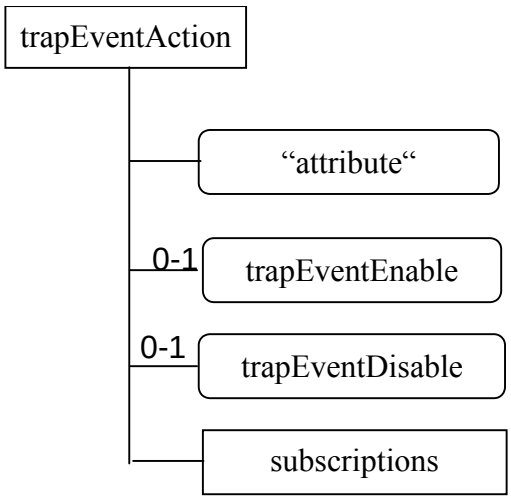


Figure B.15: Structure of the trapEventAction resource

The trapEventAction resource shall contain the following sub resource:

Table B.30

SubResource	Mandatory/Optional	Multiplicity	Description
subscriptions	M	1	See clause 9.2.3.22.

and the following attributes:

Table B.31

AttributeName	Mandatory/Optional	Type	Description
accessRightID	M	RW	See clause 9.2.2.
creationTime	M	RO	See clause 9.2.2.
lastModifiedTime	M	RO	See clause 9.2.2.
trapEventEnable	O	RW	The action that enables the Trap mode.
trapEventDisable	O	RW	The action that disables the Trap mode.

The execution of an action is triggered by the UPDATE of the corresponding action attribute.

A subsequent action shall not be authorized before the completion of the previous one.

B.2.7 Resource etsiPerformanceLog

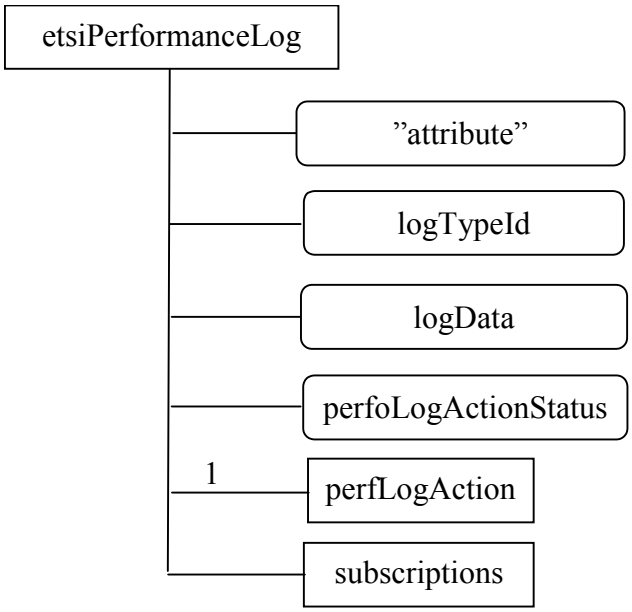


Figure B.16: Structure of the etsiPerformanceLog resource

The etsiPerformanceLog resource shall contain the following sub resources:

Table B.32

SubResource	Mandatory/Optional	Multiplicity	Description
perfLogAction	M	1	A sub-resource that contains the action to be executed.
subscriptions	M	1	See clause 9.2.3.22.

and the following attributes:

Table B.33

AttributeName	Mandatory/Optional	Type	Description
expirationTime	M	RW	See clause 9.2.2.
accessRightID	M	RW	See clause 9.2.2.
searchStrings	M	RW	See clause 9.2.2.
creationTime	M	RO	See clause 9.2.2.
lastModifiedTime	M	RO	See clause 9.2.2.
moID	M	WO	See clause 9.2.3.27.
originalMO	M	WO	See clause 9.2.3.27.
logTypeId	M	RW	Identifies the type of log that is concerned by the action.
logData	M	R	Diagnostic data logged upon event of interests defined by this diagnostic function. The format of LogData is implementation dependent. It could be an XML file, a JSON document, or any other encoding format.
perfoLogActionStatus	M	RO	Indicates the status of the Action (including a progress indicator, a final state and a reminder of the requested action).

B.2.7.1 Resource perfLogAction

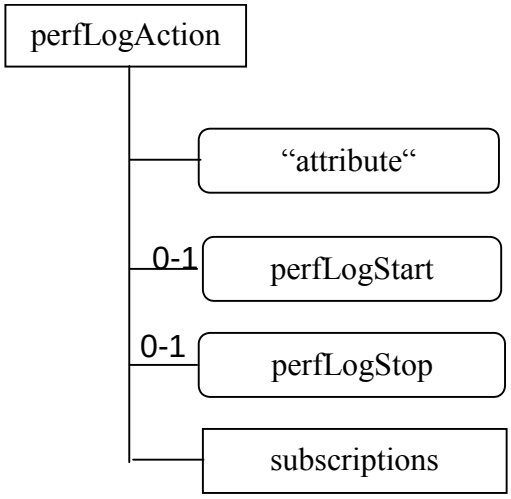


Figure B.17: Structure of the perfLogAction resource

The perfLogAction resource shall contain the following sub resource:

Table B.34

SubResource	Mandatory/Optional	Multiplicity	Description
subscriptions	M	1	See clause 9.2.3.22.

and the following attributes:

Table B.35

AttributeName	Mandatory/Optional	Type	Description
accessRightID	M	RW	See clause 9.2.2.
creationTime	M	RO	See clause 9.2.2.
lastModifiedTime	M	RO	See clause 9.2.2.
perfLogStart	O	RW	The action that allows to start the log corresponding to the mentioned logTypeId.
perfLogStop	O	RW	The action that allows to stop the log corresponding to the mentioned logTypeId.

The execution of an action is triggered by the UPDATE of the corresponding action attribute.

A subsequent action shall not be authorized before the completion of the previous one.

B.2.8 Resource etsiFirmware

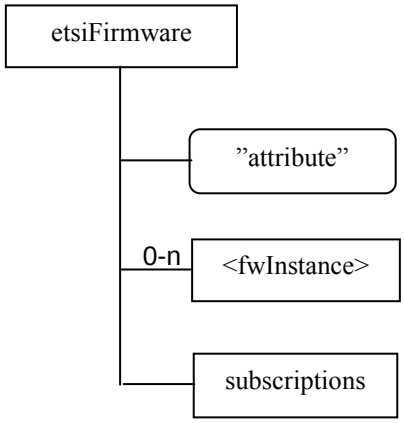


Figure B.18: Structure of the etsiFirmware resource

The etsiFirmware resource shall contain the following sub resources:

Table B.36

SubResource	Mandatory/Optional	Multiplicity	Description
<fwInstance>	O	0.n	The sub-resource that contains the information about one firmware of the device.
subscriptions	M	1	See clause 9.2.3.22.

and the following attributes:

Table B.37

AttributeName	Mandatory/Optional	Type	Description
expirationTime	M	RW	See clause 9.2.2.
accessRightID	M	RW	See clause 9.2.2.
searchStrings	M	RW	See clause 9.2.2.
creationTime	M	RO	See clause 9.2.2.
lastModifiedTime	M	RO	See clause 9.2.2.
moID	M	WO	See clause 9.2.3.27.
originalMO	M	WO	See clause 9.2.3.27.
description	O	RW	The text-format description of mgmtObj.

B.2.8.1 Resource <fwInstance>

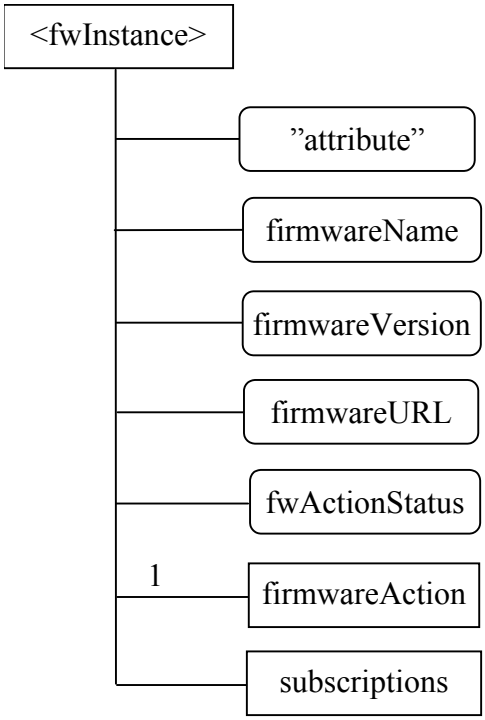


Figure B.19: Structure of the <fwInstance> resource

The <fwInstance> resource shall contain the following sub resources:

Table B.38

SubResource	Mandatory/Optional	Multiplicity	Description
firmwareAction	M	1	A sub-resource that contains the actions to be executed.
subscriptions	M	1	See clause 9.2.3.22.

and the following attributes:

Table B.39

AttributeName	Mandatory/Optional	Type	Description
accessRightID	M	RW	See clause 9.2.2.
creationTime	M	RO	See clause 9.2.2.
lastModifiedTime	M	RO	See clause 9.2.2.
originalMO	M	WO	See clause 9.2.3.27.
firmwareName	M	RW	The name of the firmware to be used on the device.
firmwareVersion	M	RW	The version of the concerned firmware.
firmwareURL	M	RW	The URL from which the firmware image can be downloaded.
fwActionStatus	M	RO	Indicates the status of the Action (including a progress indicator, a final state and a reminder of the requested action).

B.2.8.2 Resource firmwareAction

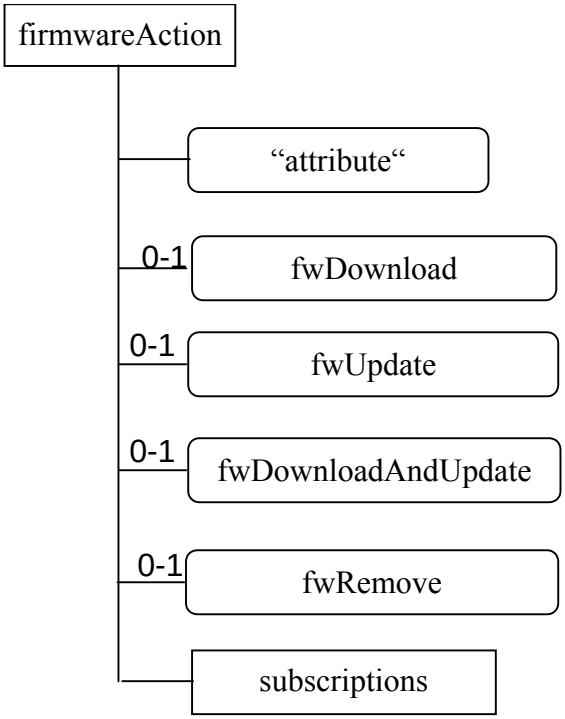


Figure B.20: Structure of the firmwareAction resource

The firmwareAction resource shall contain the following sub resource:

Table B.40

SubResource	Mandatory/Optional	Multiplicity	Description
subscriptions	M	1	See clause 9.2.3.22.

and the following attributes:

Table B.41

AttributeName	Mandatory/Optional	Type	Description
accessRightID	M	RW	See clause 9.2.2.
creationTime	M	RO	See clause 9.2.2.
lastModifiedTime	M	RO	See clause 9.2.2.
fwDownload	O	RW	The action that allows to download a new firmware.
fwUpdate	O	RW	The action that allows to install a new firmware previously downloaded.
fwDownloadAndUpdate	O	RW	The action that allows to download then install a new firmware in a single operation.
fwRemove	O	RW	The action that allows to remove a firmware package previously downloaded.

The execution of an action is triggered by the UPDATE of the corresponding action attribute.

A subsequent action shall not be authorized before the completion of the previous one.

B.2.9 Resource etsiSoftware

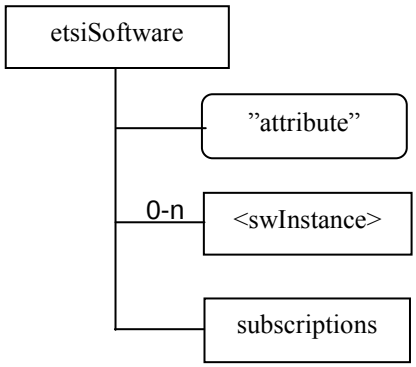


Figure B.21: Structure of the etsiSoftware resource

The etsiSoftware resource shall contain the following sub resources:

Table B.42

SubResource	Mandatory/Optional	Multiplicity	Description
<swInstance>	O	0..n	The sub-resource that contains the information about one piece of software of the device.
subscriptions	M	1	See clause 9.2.3.22.

and the following attributes:

Table B.43

AttributeName	Mandatory/Optional	Type	Description
expirationTime	M	RW	See clause 9.2.2.
accessRightID	M	RW	See clause 9.2.2.
searchStrings	M	RW	See clause 9.2.2.
creationTime	M	RO	See clause 9.2.2.
lastModifiedTime	M	RO	See clause 9.2.2.
moID	M	WO	See clause 9.2.3.27.
originalMO	M	WO	See clause 9.2.3.27.
description	O	RW	The text-format description of mgmtObj.

B.2.9.1 Resource <swInstance>

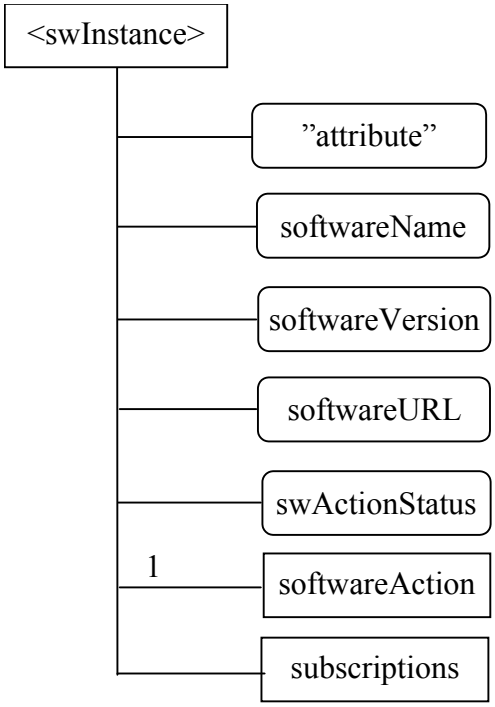


Figure B.22: Structure of the <swInstance> resource

The <swInstance> resource shall contain the following sub resources:

Table B.44

SubResource	Mandatory/Optional	Multiplicity	Description
softwareAction	M	1	A sub-resource that contains the action to be executed.
subscriptions	M	1	See clause 9.2.3.22.

and the following attributes:

Table B.45

AttributeName	Mandatory/Optional	Type	Description
accessRightID	M	RW	See clause 9.2.2.
creationTime	M	RO	See clause 9.2.2.
lastModifiedTime	M	RO	See clause 9.2.2.
originalMO	M	WO	See clause 9.2.3.27.
softwareName	M	RW	The name of the software.
softwareVersion	M	RW	The version of the software.
softwareURL	M	RW	The URL of the software to be downloaded.
swActionStatus	M	RO	Indicates the status of the Action (including a progress indicator, a final state and a reminder of the requested action).

B.2.9.2 Resource softwareAction

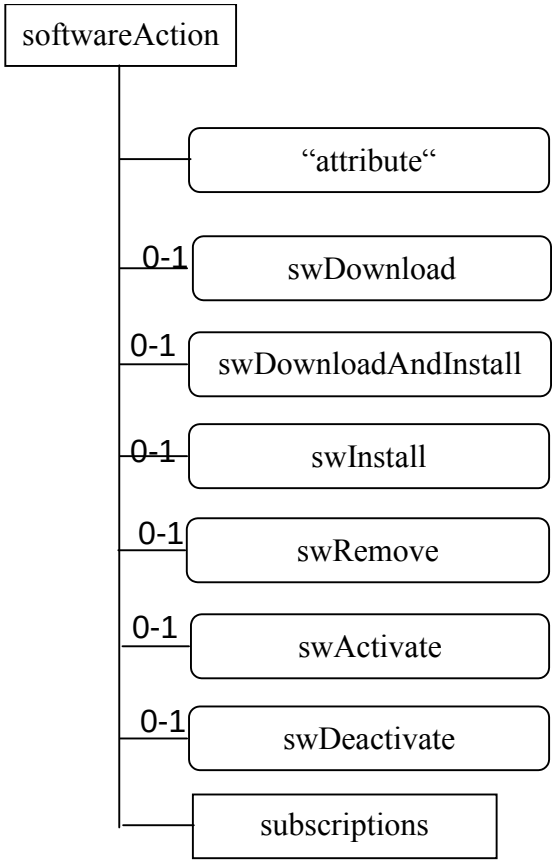


Figure B.23: Structure of the softwareAction resource

The softwareAction resource shall contain the following sub resource:

Table B.46

SubResource	Mandatory/Optional	Multiplicity	Description
subscriptions	M	1	See clause 9.2.3.22.

and the following attributes:

Table B.47

AttributeName	Mandatory/Optional	Type	Description
accessRightID	M	RW	See clause 9.2.2.
creationTime	M	RO	See clause 9.2.2.
lastModifiedTime	M	RO	See clause 9.2.2.
swDownload	O	RW	The action that allows to download a new software.
swDownloadAndInstall	O	RW	The action that allows to download then install a new software in a single operation.
swInstall	O	RW	The action that allows to install a software. previously downloaded.
swRemove	O	RW	The action that allows to remove a software package previously downloaded.
swActivate	O	RW	The action that allows to activate a software previously installed.
swDeactivate	O	RW	The action that allows to deactivate a software.

The execution of an action is triggered by the UPDATE of the corresponding action attribute.

A subsequent action shall not be authorized before the completion of the previous one.

B.2.10 Resource etsiReboot

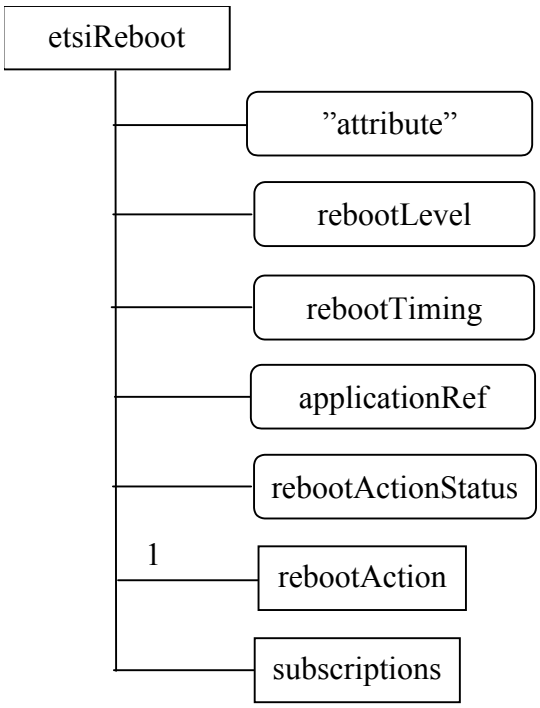


Figure B.24: Structure of the etsiReboot resource

The etsiReboot resource shall contain the following sub resources:

Table B.48

SubResource	Mandatory/Optional	Multiplicity	Description
rebootAction	M	1	A sub- resource that contains the action to be executed.
subscriptions	M	1	See clause 9.2.3.22.

and the following attributes:

Table B.49

AttributeName	Mandatory/Optional	Type	Description
expirationTime	M	RW	See clause 9.2.2.
accessRightID	M	RW	See clause 9.2.2.
searchStrings	M	RW	See clause 9.2.2.
creationTime	M	RO	See clause 9.2.2.
lastModifiedTime	M	RO	See clause 9.2.2.
moID	M	WO	See clause 9.2.3.27.
originalMO	M	WO	See clause 9.2.3.27.
rebootLevel	M	RW	Specify the level of rebooting (e.g. OS level, all applications, a specific application).
rebootTiming	M	RW	Specify the timing requirement of rebooting (e.g. immediately, gracefully).
applicationRef	M	RW	Indicates which application has to be rebooted
rebootActionStatus	M	RO	Indicates the status of the Action (including a progress indicator, a final state and a reminder of the requested action).

B.2.10.1 Resource rebootAction

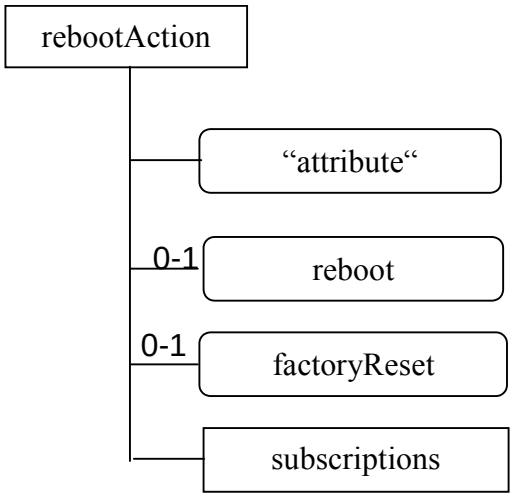


Figure B.25: Structure of the rebootAction resource

The rebootAction resource shall contain the following sub resource:

Table B.50

SubResource	Mandatory/Optional	Multiplicity	Description
subscriptions	M	1	See clause 9.2.3.22.

and the following attributes:

Table B.51

AttributeName	Mandatory/Optional	Type	Description
accessRightID	M	RW	See clause 9.2.2.
creationTime	M	RO	See clause 9.2.2.
lastModifiedTime	M	RO	See clause 9.2.2.
reboot	O	RW	The action that allows to reboot the device.
factoryReset	O	RW	The action that allows to make the device returning to the factory settings.

The execution of an action is triggered by the UPDATE of the corresponding action attribute.

A subsequent action shall not be authorized before the completion of the previous one.

B.2.11 Resource etsiAreaNwkInfo

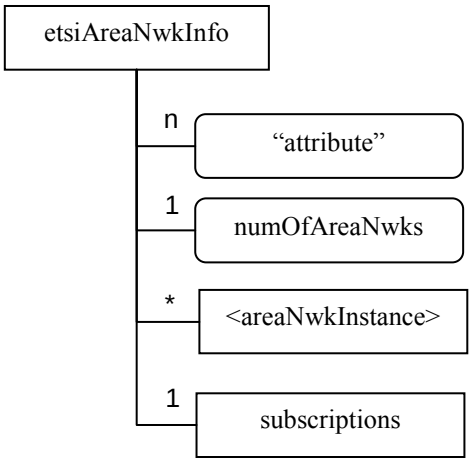


Figure B.26: Structure of the etsiAreaNwkInfo resource

The etsiAreaNwkInfo resource shall contain the following sub resources:

Table B.52

SubResource	Mandatory/Optional	Multiplicity	Description
<areaNwkInstance>	M	0::n	The sub-resource that contains the information about one of the M2M area networks connecting to the same M2M GW.
subscriptions	M	1	See clause 9.2.3.22.

and the following attributes:

Table B.53

AttributeName	Mandatory/Optional	Type	Description
expirationTime	M	RW	See clause 9.2.2.
accessRightID	M	RW	See clause 9.2.2.
searchStrings	M	RW	See clause 9.2.2.
creationTime	M	RO	See clause 9.2.2.
lastModifiedTime	M	RO	See clause 9.2.2.
moID	M	WO	See clause 9.2.3.27.
originalMO	M	WO	See clause 9.2.3.27.
description	O	RW	The text-format description of MO.
numOfAreaNwks	M	RO	The number of area networks attached with the M2M GW.

B.2.11.1 Resource <areaNwkInstance>

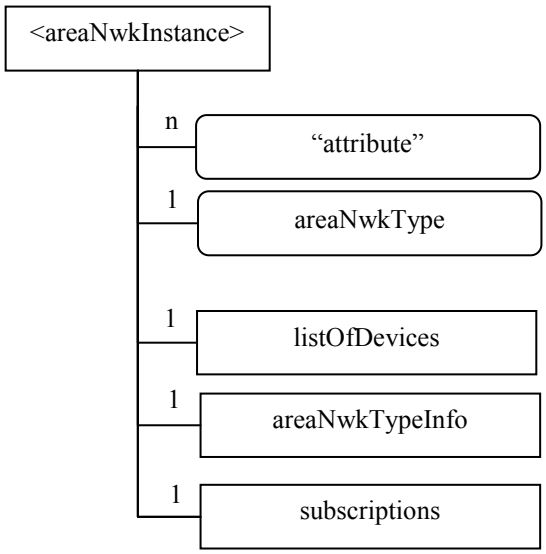


Figure B.27: Structure of the <areaNwkInstance> resource

The <areaNwkInstance> resource shall contain the following sub resources:

Table B.54

SubResource	Mandatory/Optional	Multiplicity	Description
listOfDevices	M	1	It is a group instance as defined in current M2M functional architecture. Each member of listOfDevices is actually a reference to a corresponding <GSCL>/attachedDevices/<attachedDevice>.
areaNwkTypeInfo	M	1	It is the placeholder to contain parameters specific to a type of M2M area networks, as denoted by the attribute, "areaNwkType". For example, if areaNwkType shows that the M2M area network is 6LoWPAN, areaNwkTypeInfo will contain 6LoWPAN-specific parameters. It will at least contain an "addressType" attribute to show the address type of this M2M area network. It could be IPv4, IPv6, non-IP, etc.
subscriptions	M	1	See clause 9.2.3.22.

and the following attributes:

Table B.55

AttributeName	Mandatory/Optional	Type	Description
expirationTime	M	RW	See clause 9.2.2.
accessRightID	M	RW	See clause 9.2.2.
searchStrings	M	RW	See clause 9.2.2.
creationTime	M	RO	See clause 9.2.2.
lastModifiedTime	M	RO	See clause 9.2.2.
originalMO	M	WO	See clause 9.2.3.27.
description	O	RW	The text-format description of this MO.
areaNwkType	M	RW	The type of this area network instance, such as bluetooth/6LoWPAN, 802.15.4/6LoWPAN, WiFi, PLC, etc.

B.2.12 Resource etsiAreaNwkDeviceInfo

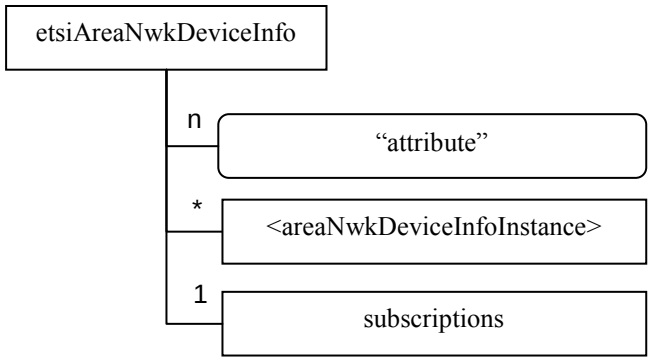


Figure B.28: Structure of the etsiAreaNwkDeviceInfo resource

The etsiAreaNwkDeviceInfo resource shall contain the following sub resources:

Table B.56

SubResource	Mandatory/Optional	Multiplicity	Description
<areaNwkDeviceInfoInstance>	M	1:n	A device may associate with multiple area networks, but at least one. This sub-resource contains parameters and information of the device related to each area network.
subscriptions	M	1	See clause 9.2.3.22.

and the following attributes:

Table B.57

AttributeName	Mandatory/Optional	Type	Description
expirationTime	M	RW	See clause 9.2.2.
accessRightID	M	RW	See clause 9.2.2.
searchStrings	M	RW	See clause 9.2.2.
creationTime	M	RO	See clause 9.2.2.
lastModifiedTime	M	RO	See clause 9.2.2.
originalMO	M	WO	See clause 9.2.3.27.
description	O	RW	The text-format description of this MO.

B.2.12.1 Resource <areaNwkDeviceInfoInstance>

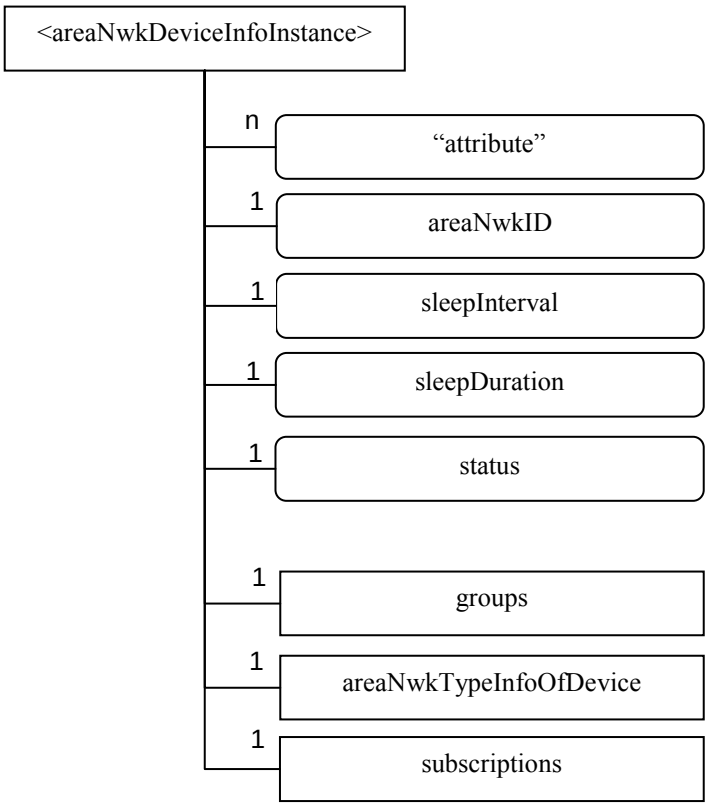


Figure B.29: Structure of the <areaNwkDeviceInfoInstance> resource

The areaNwkDeviceInfoInstance resource shall contain the following sub resources:

Table B.58

SubResource	Mandatory/Optional	Multiplicity	Description
groups	M	1	It is the standard *groups* resource as defined in ETSI M2M functional architecture document and has following two group instances: • listOfDeviceNeighbors: each member of this group stands for a device which is the neighbour of the same <attachedDevice> • listOfDeviceApplications: each member of this group stands for a DA on the same <attachedDevice> using the same M2M area network.
areaNwkTypeInfoOfDevice	M	1	It is the placeholder to contain parameters specific to a type of M2M area networks, as denoted by the attribute, "areaNwkType". For example, if areaNwkType shows that the M2M area network is 6LoWPAN, areaNwkTypeInfo will contain 6LoWPAN-specific parameters of the device.
subscriptions	M	1	See clause 9.2.3.22.

and the following attributes:

Table B.59

AttributeName	Mandatory/Optional	Type	Description
expirationTime	M	RW	See clause 9.2.2.
accessRightID	M	RW	See clause 9.2.2.
searchStrings	M	RW	See clause 9.2.2.
creationTime	M	RO	See clause 9.2.2.
lastModifiedTime	M	RO	See clause 9.2.2.
originalMO	M	WO	See clause 9.2.3.22.
description	O	RW	The text-format description of this MO.
areaNwkID	M	RW	The reference to an <areaNwkInstance> which this device associates with.
sleepInterval	M	RW	The interval between two sleeps.
sleepDuration	M	RW	The time duration of each sleep.
status	M	RW	The status of the device (sleeping or waked up).

Annex C (normative): Optional incorporation of Integrity Validation in SCL Registration

C.1 Introduction

This annex applies only when M2M Devices/Gateways and/or NSCL networks support and enable optional Integrity Validation (IVal). It describes IVal procedures which shall be performed prior to and during registering SCLs in such cases.

NOTE: Alternative implementations of the IVal capability in future releases are not precluded.

C.2 Pre-Requisites

For M2M Devices/Gateways that support IVal, it is required that prior to SCL Registration, the following conditions shall be met:

- An M2M Root Key (Kmr), or a temporary credential, is already established at the M2M Device/Gateway and the MAS in the target M2M system.
- The M2M Device/Gateway SCL that performs the request for the SCL Registration procedure and the corresponding Reference Points in the M2M Device/Gateway has been Integrity-Validated. Integrity Validation (IVal) of at least DSEC/GSEC shall be performed by a dedicated function which is not dependent on the integrity of DSEC/GSEC. DSEC/GSEC may then be used to perform IVal of other SCs and of the M2M Reference Points of the M2M Device/Gateway, as specified in clauses 5.3.6 and 5.4.6.
- The requirements for IVal specified herein for M2M Service Layer shall apply whether IVal of M2M Service-Layer functions is performed as part of a general secure boot process for the M2M Device/Gateway, or is performed as part of M2M Service-Layer procedures.

NOTE 1: Authentication uses the MAS because at this stage Kmr or the temporary credential may be the only material available for authentication, and they are known to only the MAS and the Issuer SCL (but not the Receiver SCL). The possibility of authentication using keys derived from Kmr is not excluded.

NOTE 2: Details of the interface between the Receiver SCL and the MAS are out of scope of the present document.

- A mapping of blocks of executable code which are Integrity-Validated vs. standardized SCs and communications reference-point functions shall be provisioned and securely maintained in the M2M Device/Gateway. IVal status records to be used by the local SCL and results to be reported to the NSCL shall be made on the basis of the standardized functions.

NOTE 3: The above function hides the specifics of implementation of SCL and reference points in the M2M Device/Gateway from the local SCL and the NSCL.

C.3 Procedure for Use of IVal in Registration of Issuer (Local) SCL to Receiver SCL

Executable code which is stored in physically immutable memory can be excluded from the IVal procedures.

Step 1: At the start of the authentication/authorization procedure, the M2M Device/Gateway shall perform IVal on all SCs and communications reference point blocks of executable code that are required to be Integrity Validated at this stage.

Step 2: The action taken next by the M2M Device/Gateway depends on the result of the IVal procedure, as follows:

- If all SCs and communications reference point blocks of executable code pass the IVal test, step 3 shall commence.
- If any SC or communications reference point blocks of executable code which is required for steps 3 and/or 4 or is defined by the local policy as "critical" fails the IVal test, then the M2M Device/Gateway shall prevent its SCL from performing step 3 and the authentication/authorization procedure shall be aborted.
- If any SC or communications reference point blocks of executable code which does not need to be activated for steps 3 and/or 4, and which is not defined by the local policy as "critical" fails the IVal test, then step 3 shall proceed.
- Locally present policy information shall be protected for integrity, using storage in a Secured Environment.

NOTE 1: The locally present policy described above could originate from the manufacturer or from the Service Provider but needs to be pre-provisioned to the M2M Device/Gateway by the manufacturer. It may be possible for the policy to be updated or replaced after the SCL is registered, using, e.g. OMA-DM procedures.

Step 3: The Issuer SCL and the Receiver SCL shall perform mutual authentication, using the authentication services of the MAS. The result of a successful authentication includes the MAS providing the NSCL with the Kmc (generated from Kmr) and optionally the IVal capability of the D/G, which are stored by the Receiver SCL for later use if IVal is supported at the NSCL.

NOTE 2: Shared key Kmr was established between the Issuer SCL and the MAS during M2M Service Bootstrap.

Step 4: The Issuer SCL shall send an SCL registration request, including securely time-stamped and signed IVal results to the Receiver SCL. The signature and time-stamping shall be performed by the Secured Environment which computes the IVal results. A key shall be provisioned to the Secured Environment for creating such signatures. Further details are provided in the architecture clause 5 of the present document. The procedure shall include protection against the IVal data being altered, replayed or spoofed by an unauthorized function within the M2M Device/Gateway.

Step 5: If the Receiver SCL does not support IVal, it shall accept the registration request and then shall reply to the issuer SCL. If the Receiver SCL supports IVal, it shall check the data previously received from the MAS to determine whether or not the Issuer SCL belongs to an M2M Device/Gateway which supports IVal:

- If the M2M Device/Gateway supports IVal and reports IVal results, the Receiver SCL shall perform policy-based access control to accept or reject the registration based on the reported IVal results.
- If the M2M Device/Gateway supports IVal but does not report Integrity Validation Results, this shall be interpreted as a security attack and the Receiver SCL shall reject the registration request.
- If the M2M Device/Gateway does not support IVal but reports Integrity Validation Results, this shall be interpreted as a security attack and the Receiver SCL shall reject the registration request.
- If the M2M Device/Gateway does not support IVal and does not report Integrity Validation Results, the Receiver SCL shall accept the registration request.

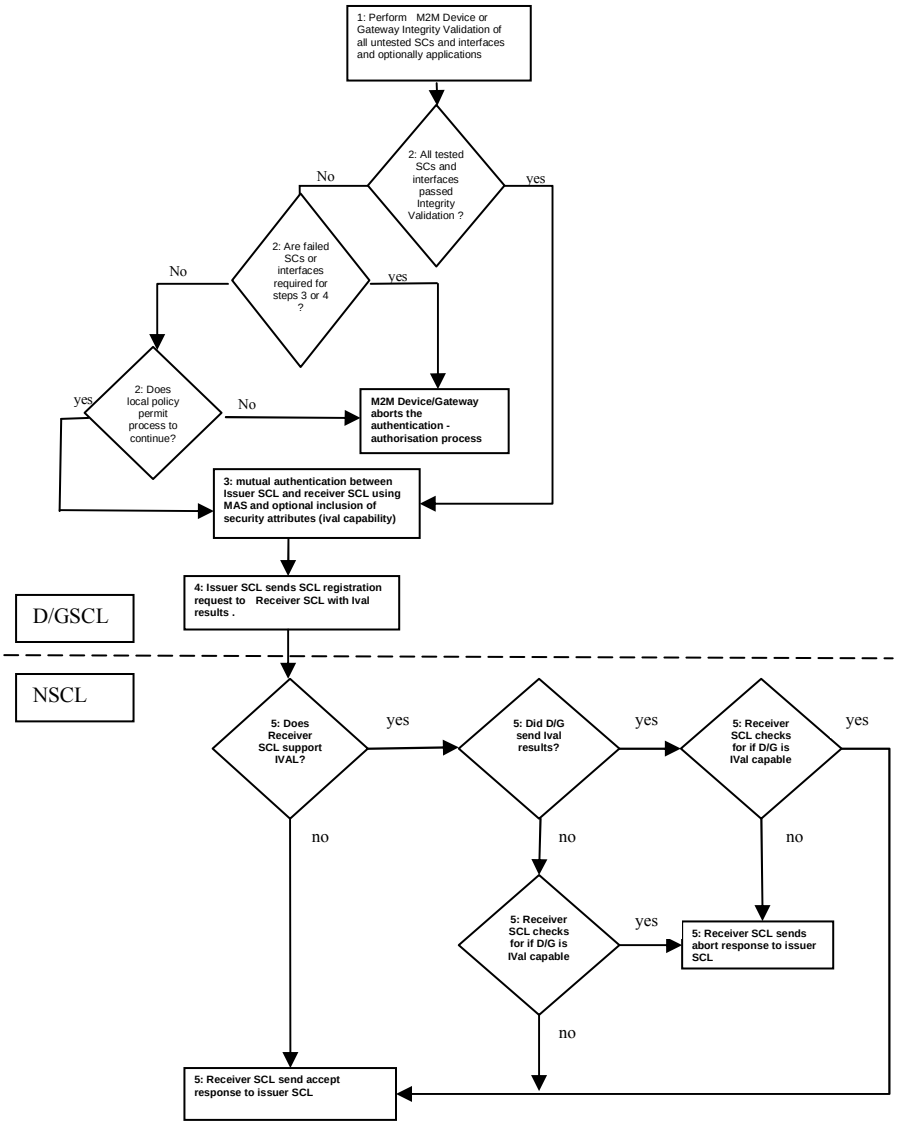


Figure C.1: Process Flow for IVal for Authentication and Authorization of Issuer SCL

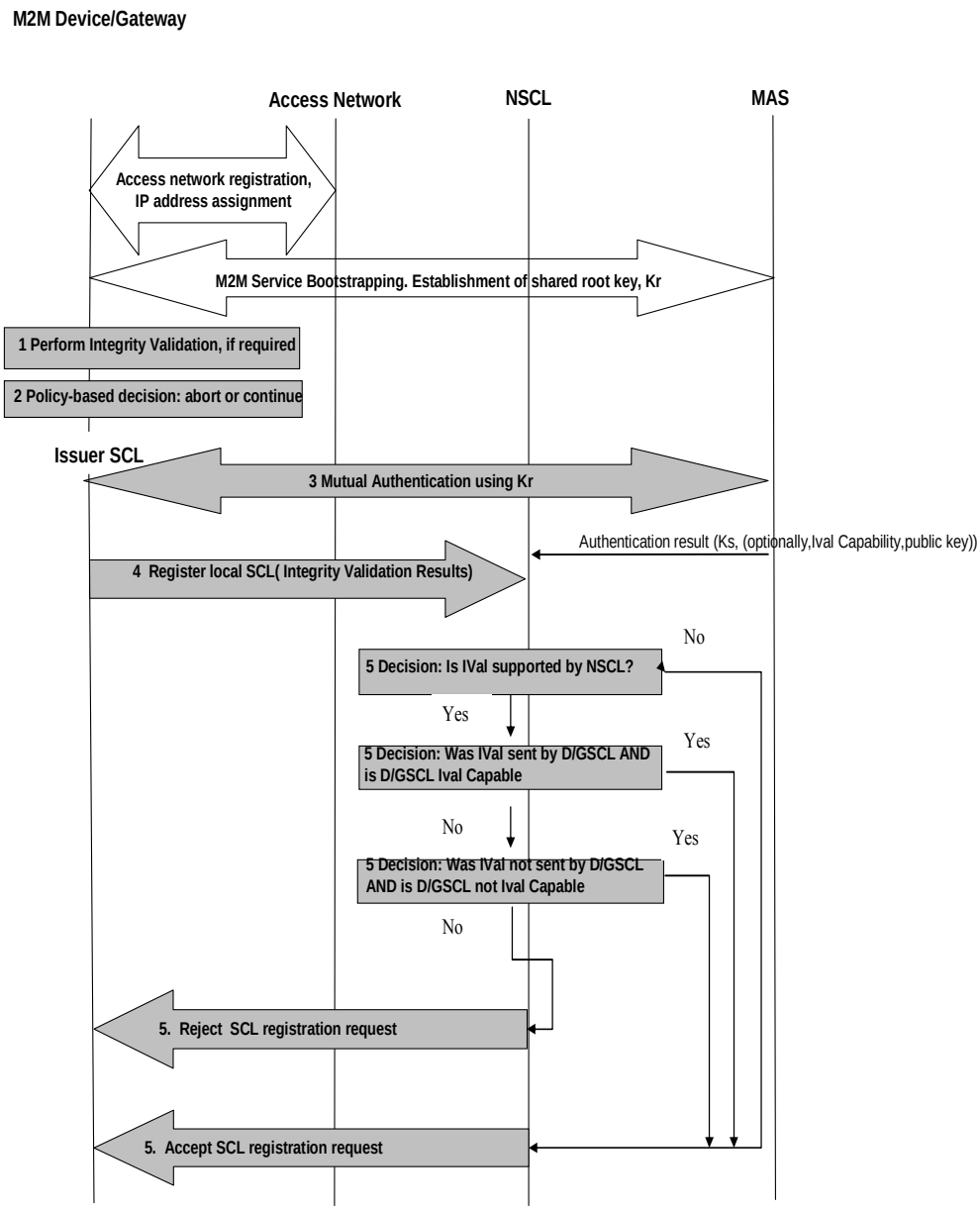


Figure C.2: Message Flow for IVal as a Pre-requisite for Authentication and Authorization of SCL Management

Annex D (normative): Interworking with XDMS

This annex is normative if and only if an XDMS is deployed for the purpose of managing the M2M resources.

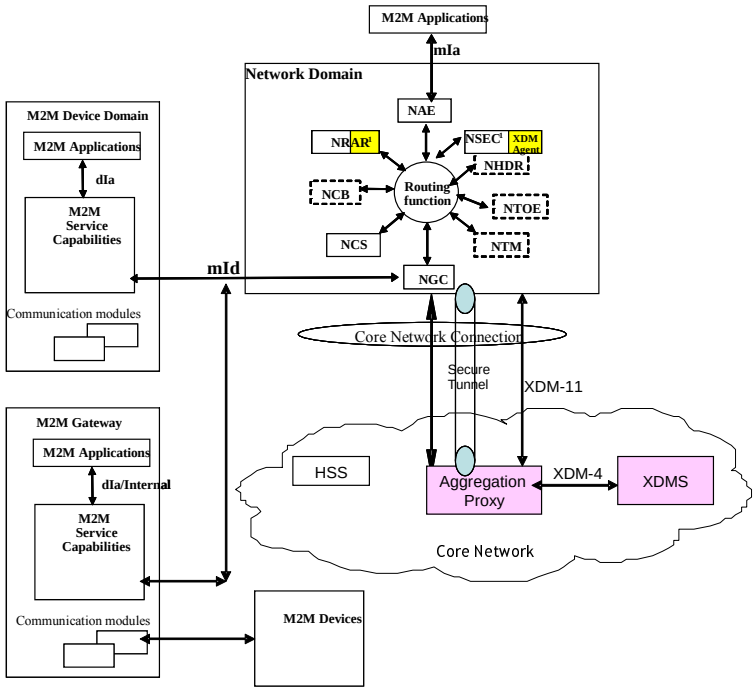
D.1 Reference Architecture for Interworking with XDMS

There are two potential options for interworking between a Network Domain and XDMS for the purpose of supporting the management of M2M resources. Below is a brief description of each option including the architectural impacts.

D.1.1 Option1: Network Domain and Core Network Nodes Owned by Different Providers

Figure D.1 shows the architecture for option 1.

In this option, the provider of the Network Domain is independent from the provider of the core network nodes which includes HSS, XDMS and the Aggregation Proxy depicted in Figure D.1.



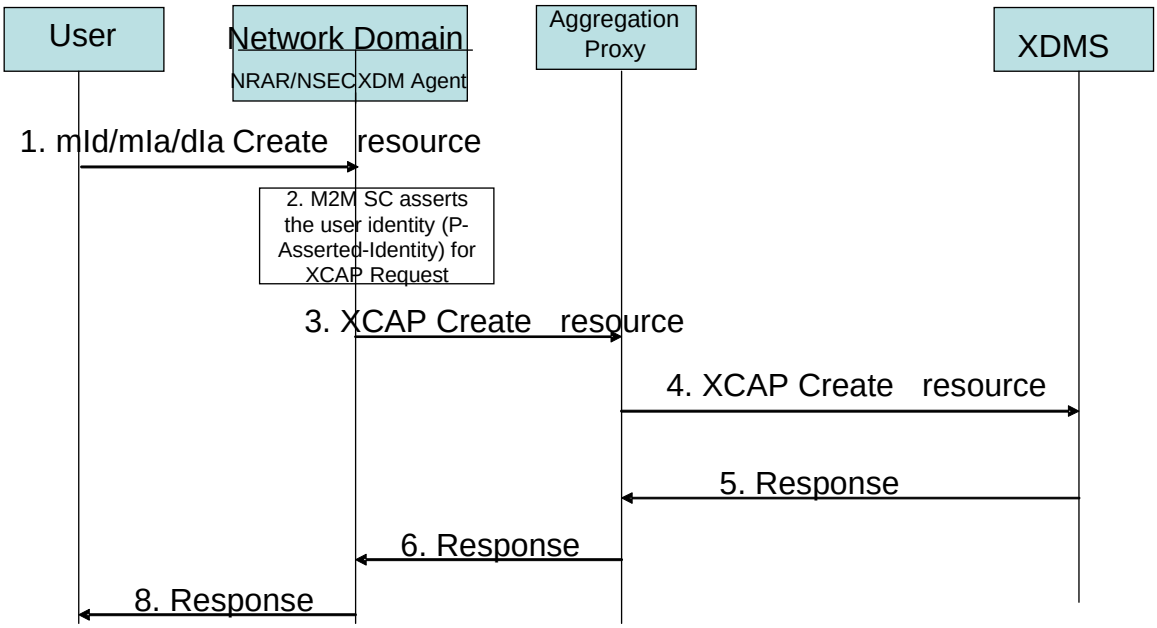
NOTE: (1) Denotes an XDM Agent as implemented in NRAR and NSEC. Typically there would be one XDM Agent per Network Domain.

Figure D.1: Option 1 - Interworking XDMS- M2M Network Domain for different service providers

In option 1, the Aggregation Proxy is the node of interaction between the two providers. A secure tunnel will have to be established between the Aggregation Proxy and the Network Domain. The means for establishing such a tunnel and indeed the exact security scheme between the two nodes is subject to offline discussion and agreement between the two service providers. As such, it is assumed in this scenario that requests arriving to the aggregation proxy have already been authorized and authenticated by the Network Domain.

Every request arriving to the Network Domain and that requires a creation/retrieval/update/deletion of a resource, results in an equivalent XCAP request created by the XDM Agent towards the Aggregation Proxy.

Figure D.2 shows a high level call flow for an exemplary case for illustrative purposes.



NOTE: It is assumed that the link between the M2M SCP and the Aggregation proxy is secured.

Figure D.2: Option 1 Example

The following is a brief description of the steps in the call flow:

Step 001: A user (D/GSCL) initiates a creation request for a new resource in the Network Domain.

Step 002: The XDM Agent creates the XCAP request equivalent to the incoming user request and inserts the proper information in the request to assert the request originator identity (P-Asserted-Identity) in the request.

Step 003: The Network Domain forwards the XCAP request to the Aggregation Proxy.

Step 004: The Aggregation Proxy in turn forwards the request to the XDMS.

Step 005: The XDMS sends back the response to the Aggregation Proxy.

Step 006: The Aggregation Proxy forwards the response to the Network Domain (the XDM Agent).

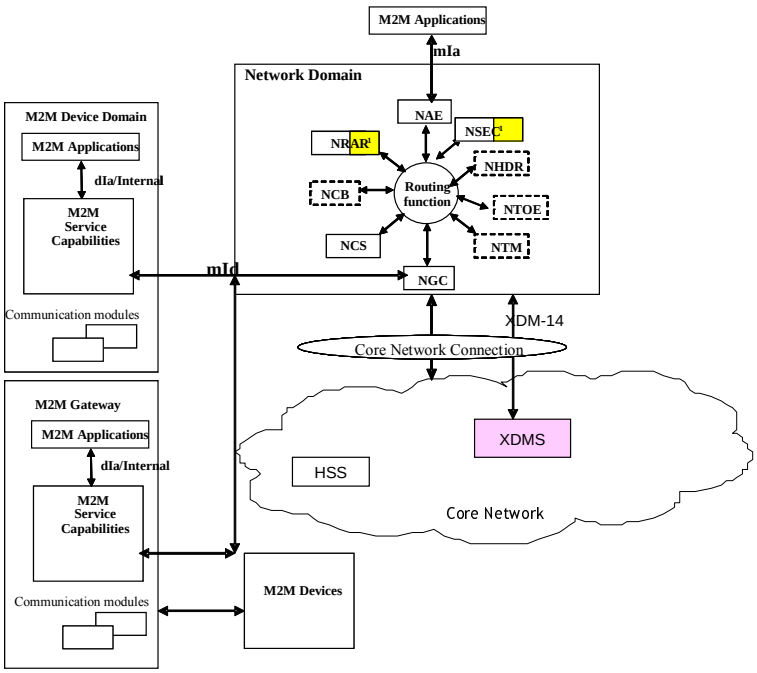
Step 007: The Network domain returns the response to the user.

D.1.2 Option2: Network Domain and Core Network Nodes Owned by the Same Provider

Figure D.3 shows the architecture for option 2.

In this option, the provider of the Network Domain is the same provider of the core network nodes which includes the HSS, and the XDMS depicted in Figure D.3.

Note that this option equally applies to the case when the Network Domain and the core network provider are independent but have an agreement that allows such an architecture to be deployed. Note in this case that the HSS may not be used by the Network Domain as opposed to the case when the 2 providers are the same.



Notes:
•In this option the authentication proxy is basically embedded in the XDM agent

NOTE: (1) Denotes an XDM Agent as implemented in NRAR and NSEC.

Figure D.3: Option 2 - Interworking XDMS- Network Domain for same service providers

Option 2 does not include an Aggregation Proxy. Rather, the XMD Agent includes an embedded authentication proxy which performs the authentication for every user (M2M device application/M2M network application) request arriving to the Network Domain.

Every request arriving to the Network Domain and that requires a creation/retrieval/update/deletion of a resource, results in an equivalent XCAP request from the XDM Agent towards the XDMS.

The XDM Agent in the Network Domain authenticates every user request requiring access to a resource. The authentication is performed against user credentials stored in HSS.

Once a user is successfully authenticated for an incoming request, the XDM Agent creates an equivalent XCAP request towards XDMS for further processing.

Note that implementations that choose to co-locate the Network Domain and the XDMS need not implement the XDM-14 interface and indeed can have an internal interface instead.

Figure D.4 shows a high level call flow for an exemplary case for illustrative purposes.

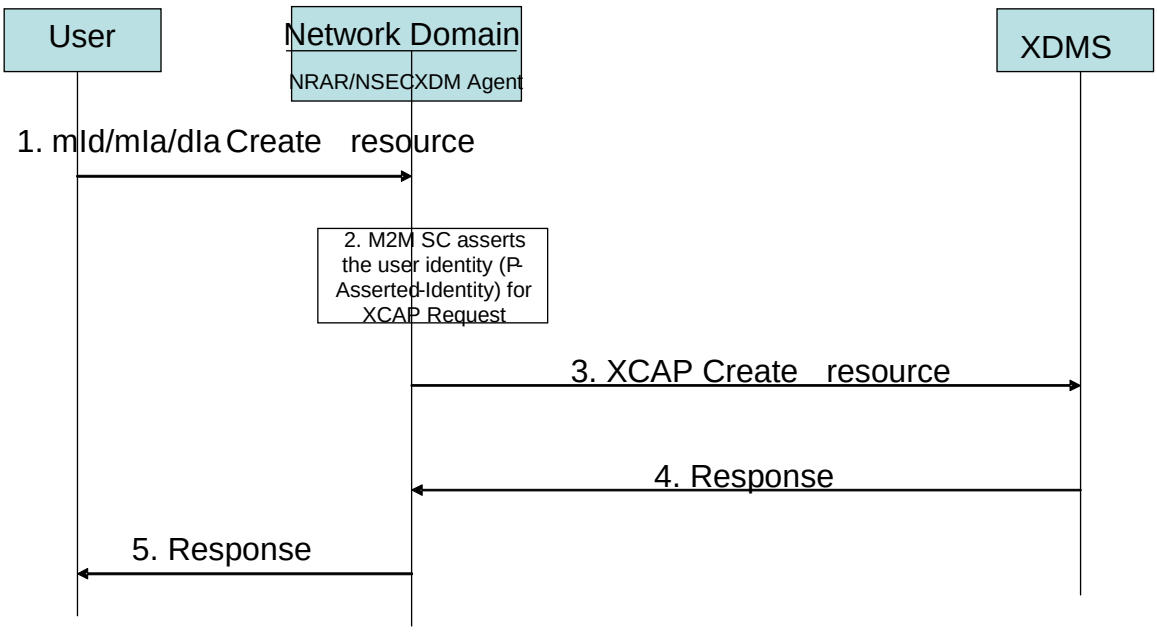


Figure D.4: Option 2 Example

The following is a brief description steps in the call flow:

Step 001: A user initiates a creation request for a new resource in the Network Domain.

Step 002: The XDM Agent creates the XCAP request equivalent to the user request and forwards it to the XDMS and inserts the P-Asserted-Identity which asserts the identity of the request originator.

Step 003: The Network Domain forwards the XCAP request to the XDMS.

Step 004: The XDMS sends back the response to the Network Domain (the XDM Agent).

Step 005: The Network Domain returns the response to the user.

D.2 Reference Points

The following is a brief overview of the purpose of the reference points:

XDM-11: The XDM-11 Reference Point supports the communication between the XDM Agent and the Aggregation Proxy. The protocol for the XDM-11 Reference Point is XCAP and XDCP.

The XDM-11 Reference Point provides the following functions:

- Management of XDM Resources (e.g. create, modify, retrieve, delete) handled by any XDMS in any network.

XDM-14: The XDM-14 Reference Point provides the following functions:

- Management of XDM Resources (e.g. create, modify, retrieve, delete) handled by a particular XDMS residing in the same network as the XDM Agent;
- Forwarding of XDM Resources handled by a particular XDMS residing in the same network as the XDM Agent.

XDM-4: The XMD-4 Reference Point provides the following functions:

- Management of XDM Resources (e.g. create, modify, retrieve, delete, restore) handled by a particular XDMS.
- History information management for XDM Documents (e.g. retrieve the History information related to an XDM Document).

- Forwarding of XDM Resources handled by a particular XDMS.
- Access Permissions management for XDM Documents handled by a particular XDMS.
- History function related preferences management (e.g. enable/disable History function) for XDM Documents handled by a particular XDMS.
- Management (e.g. create, modify, retrieve, delete, restore) of XDM Resources requested by an XDMS but handled by any other XDMS.

Annex E (normative): Reuse existing OMA-DM/BBF TR-069 for M2M REM

Remote Entity Management (REM) service capability shall be supported by all M2M SCLs. This annex describes how existing device management enablers are reused in ETSI M2M functional architecture to implement the REM service capability. In the present document, only OMA-DM [6], [8] and BBF TR-069 [10] are considered as the supporting enablers to be reused. An M2M system may choose to implement either or both of them. Other device management enablers may also be reused in a similar manner, but is out of scope.

E.1 Reference Architecture

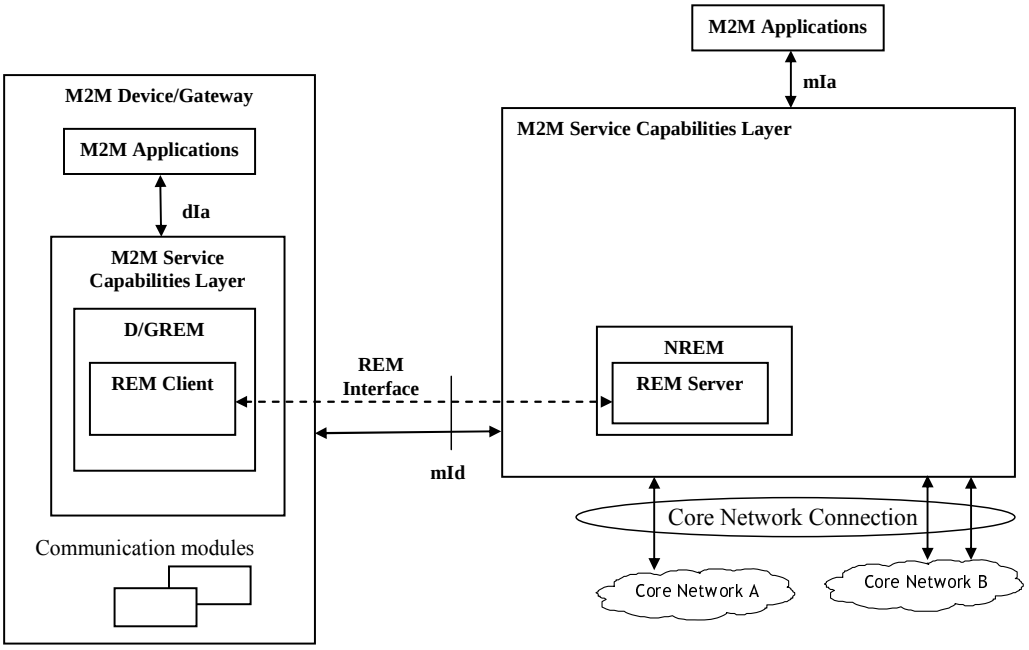


Figure E.1: Integration of device management enablers for M2M REM Service Capability

As shown in Figure E.1, the Device Management Client is integrated as a part of the D/GREM Service Capability, while the Device Management Server is integrated as a part of the NREM Service Capability.

- If OMA-DM is re-used for M2M REM Service Capability, then:
  - REM Client refers to an OMA-DM DM Client;
  - REM Server refers to an OMA-DM DM Server; and
  - REM Interface refers to DM-1 and DM-2 as defined in [6].
- If BBF TR-069 is re-used for M2M REM Service Capability, then:
  - REM Client refers to a CPE;
  - REM Server refers to an ACS; and
  - REM Interface refers to TR-069 CWMP as defined in [10].

On the M2M Device and M2M Gateway side:

- The D/GREM may collect the Management Object data from the REM Client in the local M2M Device/Gateway via an internal interface, and may create/update the corresponding <*mgmtObj*> an/or <*mgmtCmd*> resource(s) in the NSCL via the mId reference point using the RESTful access methods as described in the present document and M2M TS 102 921 [1]. Alternatively, the <*mgmtObj*> or <*mgmtCmd*> resource may be created by administrative means in the NSCL.

- Any device management activity (e.g. firmware update, fault management) in the Device/Gateway is carried out by the REM Client, communicating with a REM Server in NREM via existing Device Management interfaces.

- The GREM is also responsible for the management of the D'-type devices associated with the M2M Gateway. The Management Object data of the D'-type device is a part of the Management Object data of the M2M Gateway. The interactions between the GREM and the D'-type device for device management purpose is out of scope.

On the NSCL side:

- The NREM triggers OMA-DM [6], [8] or BBF TR-069 [10] Device Management procedures over mId resulting from actions on the <*mgmtObj*> or <*mgmtCmd*> resources by M2M Network Applications via mIa or by M2M Management Functions.

NOTE: Alternatively to REM Server being integrated as a part of the NREM a REM Server may be external to the NREM but interface with the NREM via an implementation-specific interface exposed by the REM Server.

A <*mgmtObj*> or <*mgmtCmd*> resource represents either:

- high-level management functionalities (e.g. ETSI M2M specific data model) which shall be supported by the underlying Management Object(s) on the remote entity; or

- low-level functionalities on the remote entity mapped from the data model as specified by existing device management technologies (e.g. OMA-DM [6], [8], BBF TR-069 [10]).

E.2 Reference Architecture for Managing M2M Area Networks and M2M Devices behind M2M Gateway

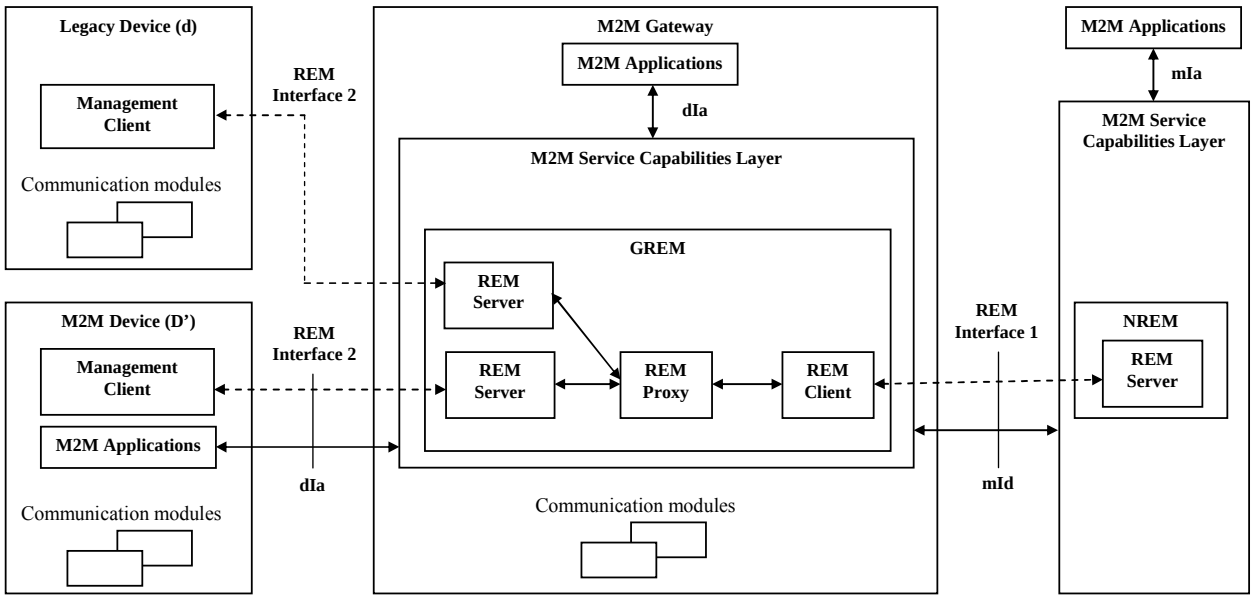


Figure E.2: Integrated Reference Architecture for Managing M2M Devices behind M2M Gateway

Figure 2 illustrates the integrated reference architecture for managing M2M Devices behind M2M Gateway, which may be legacy devices (d-type) or limited M2M Devices without M2M Service Capabilities and may not support OMA-DM [6] or BBF TR069 [10]. In order to manage M2M Devices behind M2M Gateway, GREM needs to have Server, Client and Proxy function and NREM may need additional feature in order to support the proxy function at GREM.

- If OMA-DM is reused for M2M G/NREM Service Capabilities:
  - NREM: NREM has REM Serve function, which refers to the combination of OMA-DM DM Server as defined in [6] and some additional functions for supporting GwMO as defined in [26].
  - GREM: GREM will have multiple functions including REM Client, REM Proxy, and REM Server:
    - REM Client: It refers to OMA-DM DM Client as defined in [6].
    - REM Proxy: It refers to OMA GwMO Component as defined in [26]. REM Proxy is responsible for connecting REM Client with REM Server within the GREM. It performs functions such as management protocol translation, management command conversion, management session establishment so that NREM can perform management operations on legacy d-Type devices or M2M D'-Type Devices behind M2M Gateway.
    - REM Server: It is the management server function interfacing with Management Client in M2M D'-Type Devices via REM Interface 2. REM Server is dependent on the management protocol over REM Interface 2 for M2M Gateway to manage legacy d-Type devices or M2M D'-Type Devices and the corresponding M2M Area Network which they are associated with. If there are multiple M2M Area Networks connecting to the M2M Gateway using different management protocol, GREM shall have multiple REM Servers and each REM Server is for a specific M2M Area Network.
  - M2M D' Devices: It has certain management client function in order to be managed by M2M Gateway:
    - Management Client: It communicates with REM Server in GREM via REM Interface 2. The management client is dependent on the management protocol over REM Interface 2 for M2M Gateway to manage M2M D'-Type Devices and the corresponding M2M Area Network they are associated with.
  - Legacy d-Type Devices: It has certain management client function in order to be managed by M2M Gateway:
    - Management Client: It communicates with REM Server in GREM via REM Interface 2. The management client is dependent on the management protocol over REM Interface 2 for M2M Gateway to manage legacy d-Type Devices and the corresponding M2M Area Network they are associated with.
  - REM Interface 1: It connects REM Server in NREM with REM Client in GREM. It refers to DM-1 and DM-2 interface as defined in [26].
  - REM Interface 2: It connects REM Server in GREM to Management Client in legacy d-Type Devices or M2M D'-Type Devices. It is dependent on the management protocol used for managing legacy d-Type Devices or M2M D'-Type Devices.
- If BBF-TR069 is reused for M2M G/NREM Service Capabilities:
  - NREM: NREM has REM Server function, which refers to the BBF-TR069 ACS as defined in [10].
  - GREM: It has multiple functions including REM Client, REM Proxy, and REM Server. Those functions are together referred to as CPE Proxier as defined in [10]:
    - REM Client: It refers to BBF-TR069 CPE as defined in [10].
    - REM Proxy: It refers to BBF-TR069 Proxy Module as defined in [10]. REM Proxy is responsible for connecting REM Client with REM Server within the GREM. It performs functions such as management protocol translation, management command conversion, management session establishment so that NREM can perform management operations on legacy d-Type Devices or M2M D'-Type Devices behind M2M Gateway.

        - REM Server: It refers to BBF-TR069 Control Point as defined in [10]. It is the management server function interfacing with Management Client in legacy d-Type Devices or M2M D'-Type Devices via REM Interface 2. REM Server is dependent on the management protocol over REM Interface 2 for M2M Gateway to manage legacy d-Type Devices or M2M D'-Type Devices and the corresponding M2M Area Network they are associated with. If there may be multiple M2M Area Networks connecting to the M2M Gateway using different management protocol, GREM shall have multiple REM Servers and each REM Server is for a specific M2M Area Network.

    - M2M D' Devices: It has certain management client function in order to be managed by M2M Gateway:
        - Management Client: It communicates with REM Server in GREM via REM Interface 2. The management client is dependent on the management protocol over REM Interface 2 for M2M Gateway to manage M2M D'-Type Devices and the corresponding M2M Area Network they are associated with.
    - Legacy d-Type Devices: It has certain management client function in order to be managed by M2M Gateway:
        - Management Client: It communicates with REM Server in GREM via REM Interface 2. The management client is dependent on the management protocol over REM Interface 2 for M2M Gateway to manage legacy d-Type Devices and the corresponding M2M Area Network they are associated with.
    - REM Interface 1: It connects REM Server in NREM to REM Client in GREM. It refers to CWMP protocol as defined in [10].

REM Interface 2: It connects REM Server in GREM to Management Client in legacy d-Type Devices or M2M D'-Type Devices. It is dependent on the management protocol used for managing legacy d-Type Devices or M2M D'-Type Devices.

E.3 Reference Points

mId: This reference point shall support the RESTful interface procedures as defined in clause 9.3.2.22, clause 9.3.2.23 and clause 9.3.2.24 of the present document and the corresponding parts of TS 102 921 [1], which allows the D/GREM to create <mgmtObj> and/or <mgmtCmd> resources in the NREM for the purpose of the remote entity management thereafter.

If OMA-DM technology is re-used for the M2M remote entity management, this reference point shall also support DM-1 and DM-2 interfaces as defined in [6].

- DM-1 provides an interface over which the DM Server in the NSCL may send device management notification to the DM Client on the M2M Device/Gateway to trigger the initiation of a DM session. See details in [6]. The protocol communicated over this interface is DM Notification [7].
- DM-2 provides an interface over which the DM Server in the NSCL may send device management commands to the DM Client on the M2M Device/Gateway and the DM Client may return status and alerts to the DM Server. See details in [6]. The protocol communicated over this interface can be DM session [8] or sessionless messages [9].

If BBF TR-069 technology is reused for the M2M remote entity management, this reference point shall also support TR-069 CWMP interfaces as defined in [10].

- TR-069 CWMP provides an interface over which the TR-069 ACS Server in the NSCL may send device management notification to the TR-069 CPE on the M2M Device/Gateway and the CPE may return status and alerts to the ACS Server. See details in [10]. The protocol communicated over this interface is the CPE WAN Management Protocol (CWMP).

mIa: This reference point shall support the RESTful interface procedures as defined in clause 9.3.2.22, clause 9.3.2.23 and clause 9.3.2.24 of the present document and the corresponding parts of TS 102 921 [1], which allows the NREM to handle the request from M2M Network Applications for the purpose of the remote entity management.

Annex F (informative): Pre-Configurations and Triggers for Security Procedures

Whether M2M Service Bootstrap is originally required and which option to use are pre-configured in the D/G M2M Node and in the Network M2M Node. How this pre-configuration is done is not specified by the present document.

Whether M2M Service Connection is originally required and which option to use are pre-configured in the D/G M2M Node and in the Network M2M Node. How this pre-configuration is done is not specified by the present document. If an M2M Service Connection procedure is not used, then access network security (clause 8.5 mId Security) is used for mId Security.

Figure F.1 shows the sequence of events that occur under different conditions when the D/G SCL wants to communicate to the NSCL.

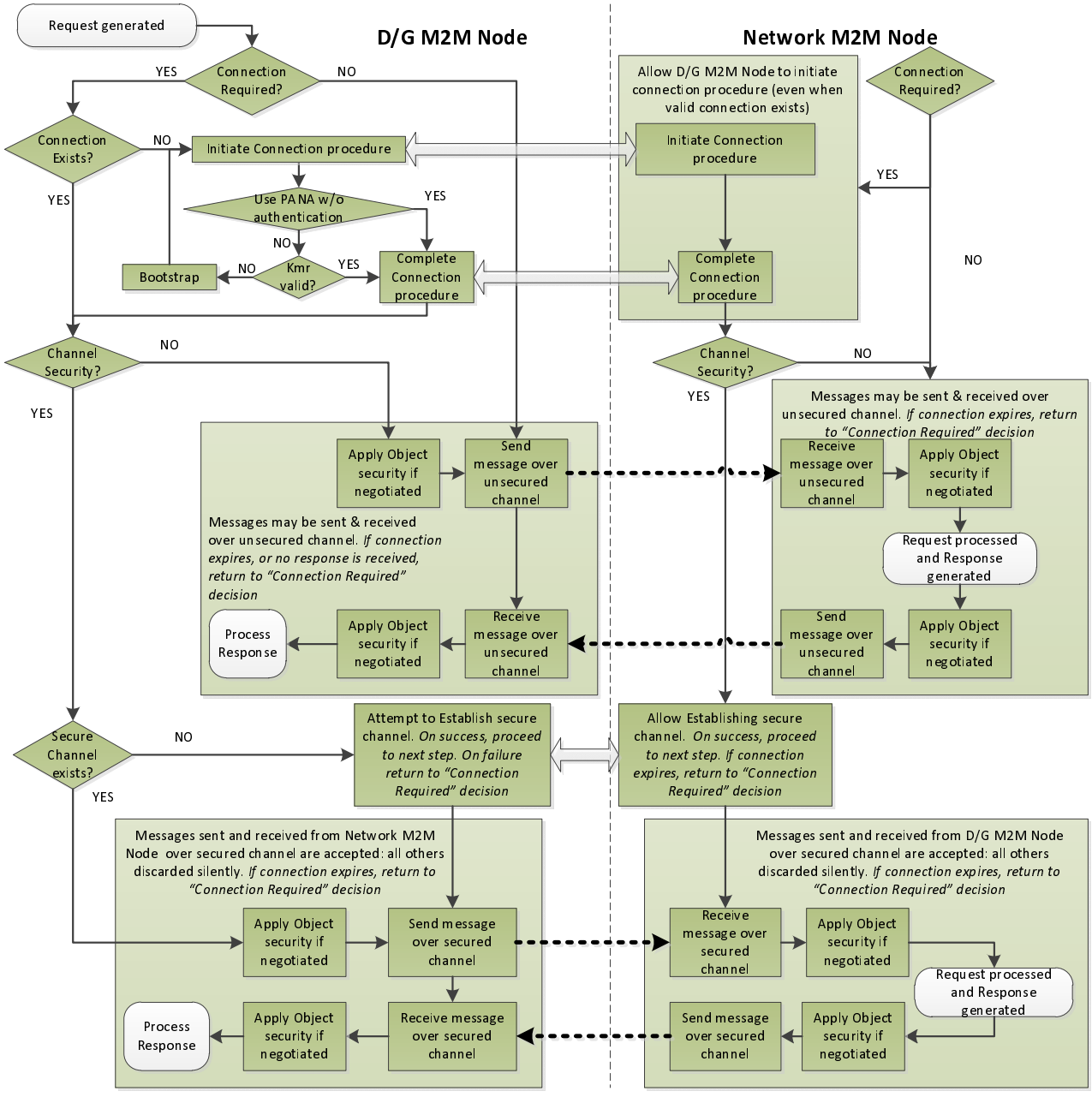


Figure F.1

The sequence of events that occur under different conditions when the D/G M2M Node wants to communicate to the Network M2M Node.

Annex G (informative): Bibliography

OMA-TS-XDM-Core-V2-120100210-D: "XML Document Management (XDM) Specification".

OMA-AD-XDM-V2-1-20091023: "XML Document Management Architecture".

OMA-ERELD-XDM-V2-1: "Enabler Release Definition for XML Document Management".

History

Document history		
V1.1.1	October 2011	Publication
V1.2.1	June 2013	Publication